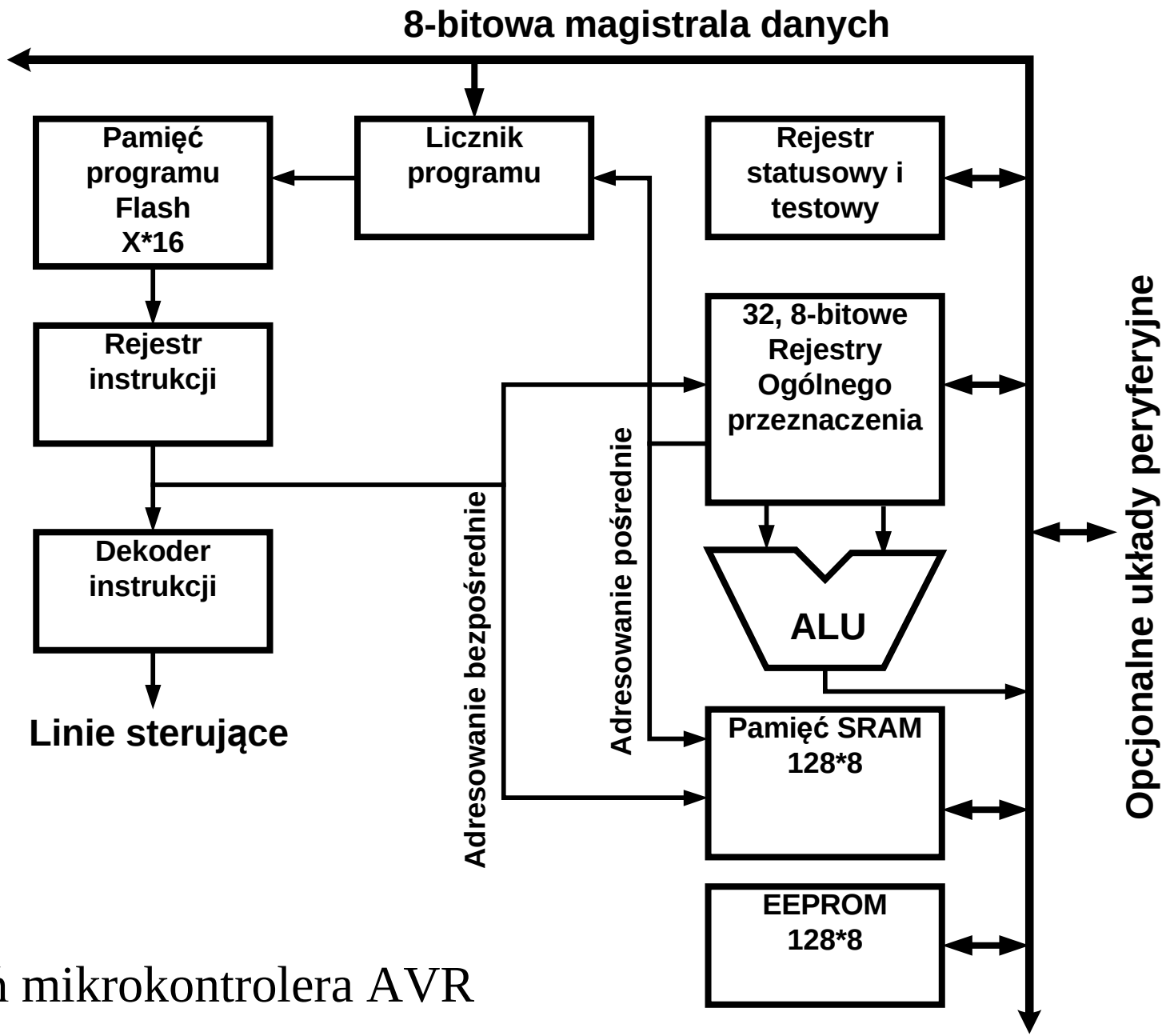
Mikrokontrolery AVR ATmega

Literatura:

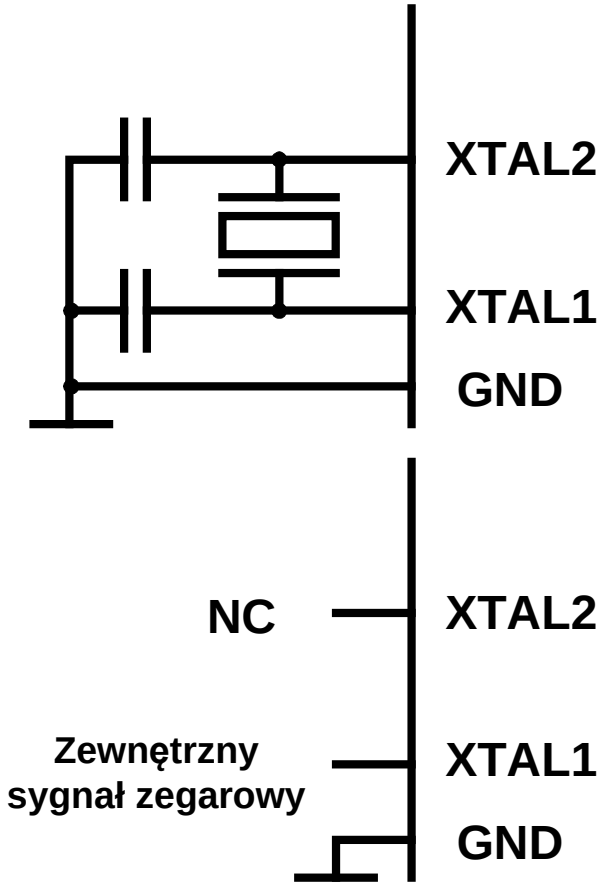
8-bit Microcontroller AVR with 32KBytes In-System Programmable Flash ATmega32 [www.atmel.com]

8-bit AVR Instruction Set [www.atmel.com]

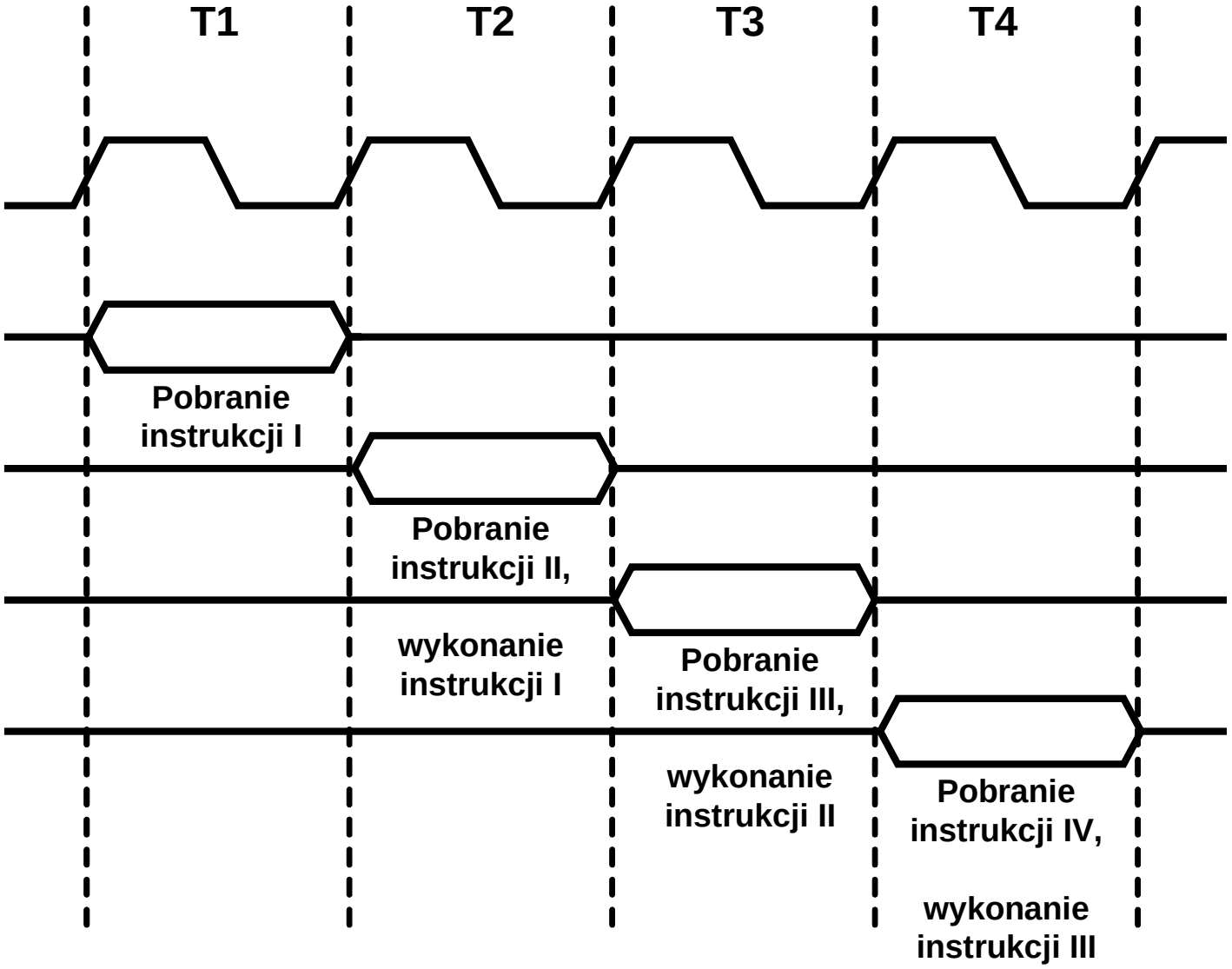
Baranowski Rafał, Mikrokontrolery AVR Atmega, BTC, Warszawa 2005



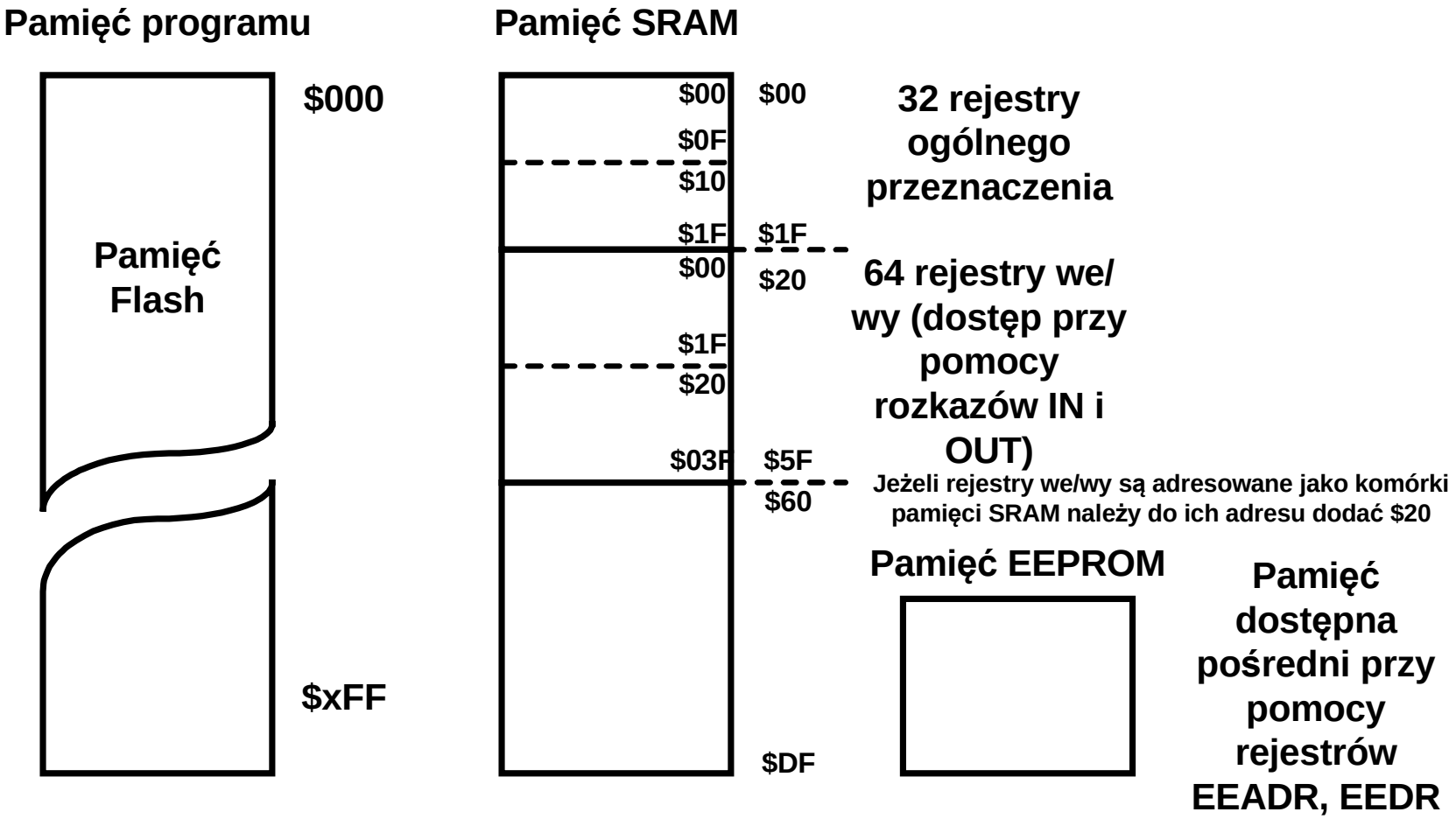
Rdzeń mikrokontrolera AVR



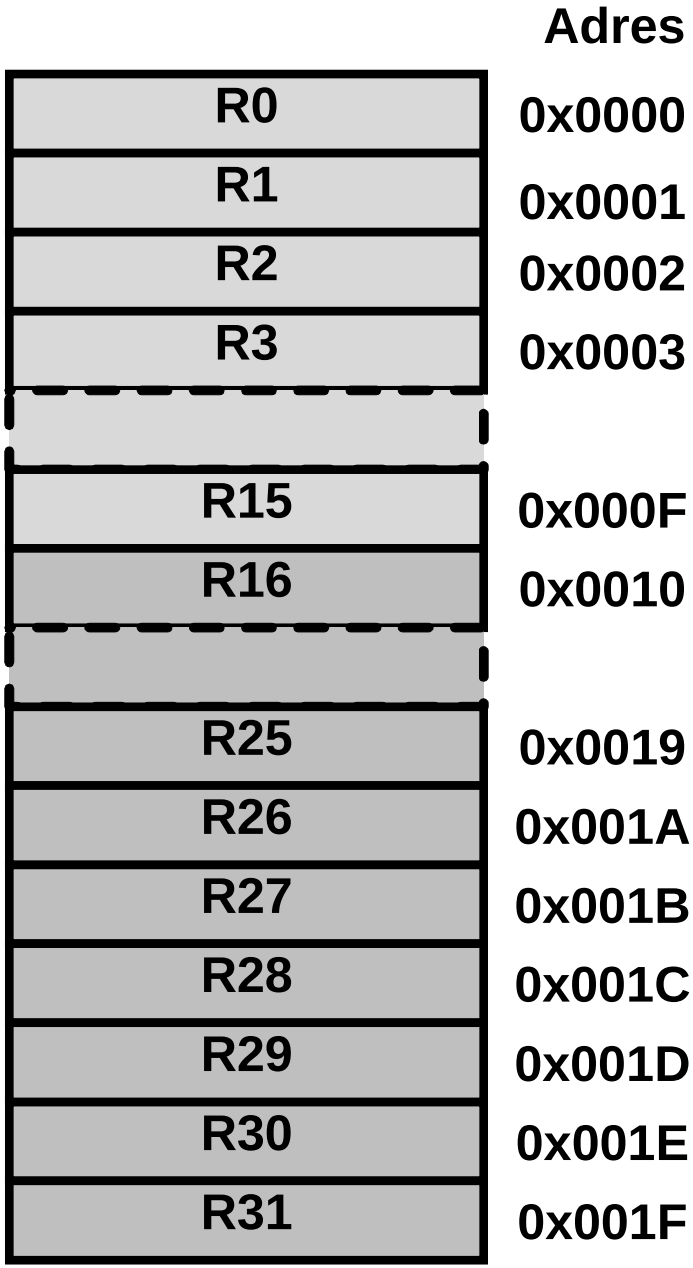
Generator zegarowy



Przetwarzanie potokowe



Organizacja pamięci mikrokontrolerów AVR



Rejestry ogólnego przeznaczenia

XL (LSB rejestru X)

XH (MSB rejestru X)

YL (LSB rejestru Y)

YH (MSB rejestru Y)

ZL (LSB rejestru Z)

ZH (MSB rejestru Z)

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

SREG

Rejestr statusu- SREG

I- globalne zezwolenie na przerwanie (odmaskowanie ***sei*** I=„1”, zamaskowanie ***cli*** I=„0”, automatycznie odtwarzana po ***reti***)

T-rejestr bitowy do kopiowania bitów

H- przeniesienie połówkowe

S- bit znaku (jeżeli nie wystąpiło V)

V- przepełnienie przy operacjach w kodzie U2

N- znacznik wartości ujemnej

Z- znacznik zera

C- przeniesienie lub pożyczka

Tryby adresacji

Adresowanie pamięci danych SRAM (pamięć SRAM, rejestry robocze, rejestry funkcyjne)

Adresowanie bezpośrednie pamięci danych

W rozkazie umieszczono dwubajtowy adres komórki pamięci np..

lds R0,0X0060; załaduj do R0 zawartość komórki pamięci o adresie 0060 hex

Adresowanie pośrednie pamięci danych

Adres umieszczono w jednym z trzech 16-to bitowych rejestrów X, Y, Z np.

ldi XH,0

ldi XL, 0x62

st X,R0; zapisz zawartość R0 do komórki pamięci o adresie 0062 hex

Adresowanie pośrednie z postinkrementacją

Po wykonaniu przesłania adres zwarty w 16-to bitowym rejestrze adresowym jest zwiększany o 1 np.

ldi XH,0

ldi XL,0x60; **X=0x0060**

ld R0,X+; zapis do rejestru R0 komórki o adresie 0060 hex

ld r1,X; zapis do rejestru R1 komórki o adresie 0061 hex

Adresowanie pośrednie z predekrementacją

Przed wykonaniem przesłania adres zwarty w 16-to bitowym rejestrze adresowym jest zmniejszany o 1 np.

ldi XH,0

ldi XL,0x63; **X=0x0063**

ld R0,-X; zapis do rejestru R0 komórki o adresie 0062 hex

ld r1,-X; zapis do rejestru R1 komórki o adresie 0061 hex

Adresowanie pośrednie z przemieszczeniem

Adres jest sumą adresu bazowego zwartego w jednym z rejestrów 16-to bitowych i przesunięcia z zakresu 0-63 np.

ldi ZH,0

ldi ZL,0x60; Z=0x060

std Z+5,R0; zapisz zawartość rejestru R0 do komórki o adresie 0065 hex

Adresowanie rejestrów roboczych

Bezpośrednie adresowanie rejestrów roboczych

Kod wybranej instrukcji zawiera adres rejestru np.

clr R1; zerowanie zawartości rejestru R1

Bezpośrednie adresowanie dwóch rejestrów roboczych

W kodzie rozkazu znajdują się adresy dwóch rejestrów np.

mov R1,R0; przesłanie zawartości rejestru R1 do R0

Adresowanie pośrednie rejestru roboczego

Adres rejestru znajduje się w 16-to bitowym rejestrze adresowym np.

ldi XH,0x00

ldi XL,0x00; adres rejestru R0 (0x0000) w X

ld R1,X

Adresowanie przestrzeni wejścia-wyjścia

Bezpośrednie adresowanie przestrzeni wejścia wyjścia jako osobnej przestrzeni adresowej

Za kodem instrukcji znajduje się adres urządzenia we-wy np.

in R1,SREG; załaduj R1 zawartością rejestru statusowego o adresie 0x3F

Bezpośrednie adresowanie przestrzeni wejścia wyjścia jako części przestrzeni adresowej pamięci danych

Za kodem rozkazu podany jest adres rejestru we-wy w przestrzenie adresowej SRAM np.

Lds R1,SREG+0x20; załaduj R1 zawartością komórki pamięci o adresie 0x3F+0x20 czyli rejestru SREG

Adresowanie pamięci programu- odczyt stałych

Tylko adresowanie pośrednie z wykorzystaniem rejestru Z np.

Stala: ,dw 0x1234 ; deklaracja stałej 2 bajtowej

ldi ZH, high(Stala*2+1)

ldi ZL, high(Stala*2+1)

lpm R0,Z; pobranie starszego bajtu stałej spod adresu Stala do R0

Możliwość adresacji z postinkrementacją

Adresowanie skoków w pamięci programu

Względne adresowanie pamięci programu

rjpm dalej; skok bez śladu

.

dalej:

Do zawartości PC dodawane jest przesunięcie od –2048 do +2047

Bezpośrednie adresowanie pamięci programu

call wykonanie; wywołanie podprogramu

.

wykonanie:

Pełny, 16-to bitowy adres skoku

Pośrednie adresowanie pamięci programu

Wykorzystuje rejestr Z w roli wskaźnika np.

ldi ZH,high(wykonaj); załadowanie starszej części adresu procedury

ldi ZL,low(wykonaj); załadowanie młodszej zcęści adresu procedury

Icall; wywołanie procedury

.

wykonaj:

.

ret

Rejestry funkcyjne

Wskaźnik stosu:

SPH (MSB)

SPL (LSB)

Typowa inicjalizacja stosu:

ldi R15,high(RAM_koniec)

out SPH,R15

ldi R15,low(RAM_koniec)

out SPL,R15

Rejestry GPIOR- rejestry ogólnego przeznaczenia zwykle trzy GPIOR0-GPIOR3. Dostępne również bitowo.

Rejestr RAMPZ, zaimplementowany jest tylko jeden rejestr RAMPZ, niezbędny do pośredniego adresowania pamięci danych w mikrokontrolerach o pamięci programu większej niż 64kB. Przy pomocy najmłodszego itu rejestru RAMPZ można wybrać stronę pamięci odczytywaną rozkazem **elpm**

Lista rozkazów

Instrukcje arytmetyczne i logiczne

Dodawanie bez CARRY

ADD Rd,Rr

Operation:

(i) Rd ← Rd + Rr

	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

Dodawanie z CARRY

ADC Rd,Rr

Operation:

(i) Rd ← Rd + Rr + C

	Syntax:	Operands:	Program Counter:
(i)	ADC Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0001	11rd	dddd	rrrr

Dodawanie danej natychmiastowej do słowa

ADIW Rd,K

Operation:

(i) Rd+1:Rd ← Rd+1:Rd + K

	Syntax:	Operands:	Program Counter:
(i)	ADIW Rd+1:Rd,K	d ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16-bit Opcode:

1001	0110	KKdd	KKKK

Odejmowanie bez CARRY

SUB Rd,Rr

Operation:

(i) Rd ← Rd - Rr

	Syntax:	Operands:	Program Counter:
(i)	SUB Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0001	10rd	dddd	rrrr

Odejmowanie danej natychmiastowej

SUBI Rd,K

Operation:

(i) Rd ← Rd - K

	Syntax:	Operands:	Program Counter:
(i)	SUBI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0101	KKKK	dddd	KKKK

Odejmowanie z CARRY

SBC Rd,Rr

Operation:

(i) Rd ← Rd - Rr - C

	Syntax:	Operands:	Program Counter:
(i)	SBC Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	10rd	dddd	rrrr

Odejmowanie danej natychmiastowej z CARRY

SBCI Rd,K

Operation:

(i) Rd ← Rd - K - C

	Syntax:	Operands:	Program Counter:
(i)	SBCI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0100	KKKK	dddd	KKKK

Odejmowanie danej natychmiastowej od słowa

SBIW Rd,K

Operation:

(i) Rd+1:Rd ← Rd+1:Rd - K

	Syntax:	Operands:	Program Counter:
(i)	SBIW Rd+1:Rd,K	d ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16-bit Opcode:

1001	0111	KKdd	KKKK

Mnożenie logiczne

AND Rd,Rr

Operation:

(i) Rd ← Rd • Rr

	Syntax:	Operands:	Program Counter:
(i)	AND Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0010	00rd	dddd	rrrr

Mnożenie logiczne z daną natychmiastową

ANDI Rd,K

Operation:

(i) Rd ← Rd • K

	Syntax:	Operands:	Program Counter:
(i)	ANDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0111	KKKK	dddd	KKKK

Suma logiczna

OR Rd,Rr

Operation:

(i) Rd ← Rd v Rr

	Syntax:	Operands:	Program Counter:
(i)	OR Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0010	10rd	dddd	rrrr

Suma logiczna z daną natychmiastową

ORI Rd,K

Operation:

(i) Rd ← Rd v K

	Syntax:	Operands:	Program Counter:
(i)	ORI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0110	KKKK	dddd	KKKK

Exclusiv OR

EOR Rd,Rr

Operation:

(i) Rd ← Rd ⊕ Rr

	Syntax:	Operands:	Program Counter:
(i)	EOR Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0010	01rd	dddd	rrrr

Complement

COM Rd

Operation:

(i) Rd ← $FF - Rd

	Syntax:	Operands:	Program Counter:
(i)	COM Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0000

Negacja

NEG Rd

Operation:

(i) Rd ← $00 - Rd

	Syntax:	Operands:	Program Counter:
(i)	NEG Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0001

Ustawienie bitów w rejestrze

SBR Rd,K

Operation:

(i) Rd ← Rd v K

	Syntax:	Operands:	Program Counter:
(i)	SBR Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0110	KKKK	dddd	KKKK

Zerowanie bitów w rejestrze

CBR Rd,K

Operation:

(i) Rd ← Rd • ($FF - K)

	Syntax:	Operands:	Program Counter:
(i)	CBR Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode: (see ANDI with K complemented)

Zwiększ zawartość rejestru o 1

INC Rd

Operation:

(i) Rd ← Rd + 1

	Syntax:	Operands:	Program Counter:
(i)	INC Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0011

Zmniejsz zawartość rejestru o 1

DEC Rd

Operation:

(i) Rd ← Rd - 1

	Syntax:	Operands:	Program Counter:
(i)	DEC Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	1010

Testuj na zero lub minus

TST Rd

Operation:

(i) Rd ← Rd • Rd

	Syntax:	Operands:	Program Counter:
(i)	TST Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode: (see AND Rd, Rd)

0010	00dd	dddd	dddd

Zeruj rejestr

CLR Rd

Operation:

(i) Rd ← Rd ⊕ Rd

	Syntax:	**Operands:**	**Program Counter:**
(i)	CLR Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

Wpisz do rejestru 0xFF

SER Rd

Operation:

(i) Rd ← $FF

	Syntax:	Operands:	Program Counter:
(i)	SER Rd	16 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1110	1111	dddd	1111

Mnożenie bez znaku

MUL Rd,Rr

Operation:

(i) R1:R0 ← Rd x Rr (unsigned ← unsigned x unsigned)

	Syntax:	Operands:	Program Counter:
(i)	MUL Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	11rd	dddd	rrrr

Mnożenie liczb ze znakiem

MULS Rd,Rr

Operation:

(i) R1:R0 ← Rd × Rr (signed ← signed × signed)

	Syntax:	Operands:	Program Counter:
(i)	MULS Rd,Rr	16 ≤ d ≤ 31, 16 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	0010	dddd	rrrr

Mnożenie liczby ze znakiem i bez znaku

MULSU Rd,Rr

Operation:

(i) R1:R0 ← Rd × Rr (signed ← signed × unsigned)

	Syntax:	Operands:	Program Counter:
(i)	MULSU Rd,Rr	16 ≤ d ≤ 23, 16 ≤ r ≤ 23	PC ← PC + 1

16-bit Opcode:

0000	0011	0ddd	0rrr

Mnożenie ułamkowe bez znaku

FMUL Rd,Rr

Operation:

(i) R1:R0 ← Rd × Rr (unsigned (1.15) ← unsigned (1.7) × unsigned (1.7))

	Syntax:	Operands:	Program Counter:
(i)	FMUL Rd,Rr	16 ≤ d ≤ 23, 16≤ r ≤ 23	PC ← PC + 1

16-bit Opcode:

0000	0011	0ddd	1rrr

Mnożenie ułamkowe ze znakiem

FMULS Rd,Rr

Operation:

(i) R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × signed (1.7))

	Syntax:	Operands:	Program Counter:
(i)	FMULS Rd,Rr	16 ≤ d ≤ 23, 16≤ r ≤ 23	PC ← PC + 1

16-bit Opcode:

0000	0011	1ddd	0rrr

Mnożenie ułamkowe ze znakiem i bez znaku

FMULSU Rd,Rr

Operation:

(i) R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × unsigned (1.7))

	Syntax:	Operands:	Program Counter:
(i)	FMULSU Rd,Rr	16 ≤ d ≤ 23, 16≤ r ≤ 23	PC ← PC + 1

16-bit Opcode:

0000	0011	1ddd	1rrr

Instrukcje skoków

Skok względny bezwarunkowy

RJMP k

Operation:

(i) PC ← PC + k + 1

	Syntax:	Operands:	Program Counter:	Stack
(i)	RJMP k	-2K ≤ k < 2K	PC ← PC + k + 1	Unchanged

16-bit Opcode:

1100	kkkk	kkkk	kkkk

Skok bezpośredni pod adres zawarty w Z

IJMP

Operation:

(i)	PC ← Z(15:0)	Devices with 16 bits PC, 128K bytes Program memory maximum.
(ii)	PC(15:0) ← Z(15:0) PC(21:16) ← 0	Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack:
(i),(ii)	IJMP	None	See Operation	Not Affected

16-bit Opcode:

1001	0100	0000	1001

Rozszerzony skok bezpośredni pod adres zawarty w Z

EIJMP

Operation:

(i) PC(15:0) ← Z(15:0)
PC(21:16) ← EIND

	Syntax:	Operands:	Program Counter:	Stack:
(i)	EIJMP	None	See Operation	Not Affected

16-bit Opcode:

1001	0100	0001	1001

Skok bezpośredni

JMP k

Operation:

(i) PC ← k

	Syntax:	Operands:	Program Counter:	Stack:
(i)	JMP k	$0 \leq k < 4M$	PC ← k	Unchanged

32-bit Opcode:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Wywołanie procedury z adresacją względną

RCALL k

Operation:

(i) PC ← PC + k + 1 Devices with 16 bits PC, 128K bytes Program memory maximum.
(ii) PC ← PC + k + 1 Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack:
(i)	RCALL k	-2K ≤ k < 2K	PC ← PC + k + 1	STACK ← PC + 1 SP ← SP - 2 (2 bytes, 16 bits)
(ii)	RCALL k	-2K ≤ k < 2K	PC ← PC + k + 1	STACK ← PC + 1 SP ← SP - 3 (3 bytes, 22 bits)

16-bit Opcode:

1101	kkkk	kkkk	kkkk

Wywołanie procedury z adresacją bezpośrednią zawartością Z

ICALL

Operation:

(i) PC(15:0) ← Z(15:0) Devices with 16 bits PC, 128K bytes Program memory maximum.

(ii) PC(15:0) ← Z(15:0) Devices with 22 bits PC, 8M bytes Program memory maximum.
PC(21:16) ← 0

	Syntax:	Operands:	Program Counter:	Stack:
(i)	ICALL	None	See Operation	STACK ← PC + 1 SP ← SP - 2 (2 bytes, 16 bits)
(ii)	ICALL	None	See Operation	STACK ← PC + 1 SP ← SP - 3 (3 bytes, 22 bits)

Wywołanie procedury z adresacją rozszerzoną bezpośrednią zawartością Z

EICALL

Operation:

(i) PC(15:0) ← Z(15:0)
PC(21:16) ← EIND

	Syntax:	Operands:	Program Counter:	Stack:
(i)	EICALL	None	See Operation	STACK ← PC + 1 SP ← SP - 3 (3 bytes, 22 bits)

16-bit Opcode:

1001	0101	0001	1001

Wywołanie procedury z adresem bezpośrednim

CALL k

Operation:

(i) PC ← k — Devices with 16 bits PC, 128K bytes Program memory maximum.

(ii) PC ← k — Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter	Stack:
(i)	CALL k	0 ≤ k < 64K	PC ← k	STACK ← PC+2 SP ← SP-2, (2 bytes, 16 bits)
(ii)	CALL k	0 ≤ k < 4M	PC ← k	STACK ← PC+2 SP ← SP-3 (3 bytes, 22 bits)

32-bit Opcode:

1001	010k	kkkk	111k
kkkk	kkkk	kkkk	kkkk

Powrót z procedury

RET

Operation:

(i) PC(15:0) ← STACK Devices with 16 bits PC, 128K bytes Program memory maximum.

(ii) PC(21:0) ← STACK Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack:
(i)	RET	None	See Operation	SP←SP + 2, (2bytes,16 bits)
(ii)	RET	None	See Operation	SP←SP + 3, (3bytes,22 bits)

16-bit Opcode:

1001	0101	0000	1000

Powrót z przerwania

RETI

Operation:

(i) PC(15:0) ← STACKDevices with 16 bits PC, 128K bytes Program memory maximum.
(ii) PC(21:0) ← STACKDevices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack
(i)	RETI	None	See Operation	SP ← SP + 2 (2 bytes, 16 bits)
(ii)	RETI	None	See Operation	SP ← SP + 3 (3 bytes, 22 bits)

16-bit Opcode:

1001	0101	0001	1000

Porównaj i pomiń jeśli równy

CPSE Rd,Rr

Operation:

(i) If Rd = Rr then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	**Operands:**
(i)	CPSE Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31

Program Counter:

PC ← PC + 1, Condition false - no skip
PC ← PC + 2, Skip a one word instruction
PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

0001	00rd	dddd	rrrr

Porównaj

CP Rd,Rr

Operation:

(i) Rd - Rr

Syntax:	**Operands:**	**Program Counter:**
(i) CP Rd,Rr	$0 \leq d \leq 31, 0 \leq r \leq 31$	PC ← PC + 1

16-bit Opcode:

0001	01rd	dddd	rrrr

Porównaj z CARRY

CPC Rd,Rr

Operation:

(i) Rd - Rr - C

	Syntax:	Operands:	Program Counter:
(i)	CPC Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0000	01rd	dddd	rrrr

Porównaj z daną natychmiastową

CPI Rd,K

Operation:

(i) Rd - K

	Syntax:	Operands:	Program Counter:
(i)	CPI Rd,K	$16 \le d \le 31, 0 \le K \le 255$	$PC \leftarrow PC + 1$

16-bit Opcode:

0011	KKKK	dddd	KKKK

Pomiń jeśli bit w rejestrze wyzerowany

SBRC Rd,b

Operation:

(i) If Rr(b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	SBRC Rr,b	0 ≤ r ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1111	110r	rrrr	0bbb

Pomiń jeśli bit w rejestrze ustawiony

SBRS Rd,b

Operation:

(i) If Rr(b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	SBRS Rr,b	0 ≤ r ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1111	111r	rrrr	0bbb

Pomiń jeśli bit w rejestrze we-wy wyzerowany

SBIC A,b

Operation:

(i) If I/O(A,b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBIC A,b	0 ≤ A ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1001	1001	AAAA	Abbb

Pomiń jeśli bit w rejestrze we-wy ustawiony

SBIS A,b

Operation:

(i) If I/O(A,b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

Syntax:

(i) SBIS A,b

Operands:

$0 \le A \le 31$, $0 \le b \le 7$

Program Counter:

PC ← PC + 1, Condition false - no skip
PC ← PC + 2, Skip a one word instruction
PC ← PC + 3, Skip a two word instruction

16-bit Opcode:

1001	1011	AAAA	Abbb

Skocz jeśli bit w rejestrze statusowym ustawiony

BRBS s,k

Operation:

(i) If SREG(s) = 1 then PC ← PC + k + 1, else PC ← PC + 1

Syntax:	Operands:	Program Counter:
(i) BRBS s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	ksss

Skocz jeśli bit w rejestrze statusowym wyzerowany

BRBC s,k

Operation:

(i) If SREG(s) = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRBC s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	kkkk

Skocz jeśli równy

BREQ k

Operation:

(i) If Rd = Rr (Z = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BREQ k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k001

Skocz jeśli różny

BRNE k

Operation:

(i) If Rd ≠ Rr (Z = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	BRNE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k001

Skocz jeśli bit CARRY ustawiony

BRCS k

Operation:

(i) If C = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRCS k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k000

Skocz jeśli bit CARRY skasowany

BRCC k

Operation:

(i) If C = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRCC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k000

Skocz jeśli większy lub równy

BRSK k

Operation:

(i) If Rd ≥Rr (C = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRSH k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k000

Skocz jeśli mniejszy

BRLO k

Operation:

(i) If Rd < Rr (C = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRLO k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k000

Skocz jeśli minus

BRMI k

Operation:

(i) If N = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRMI k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k010

Skocz jeśli plus

BRPL k

Operation:

(i) If N = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRPL k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k010

Skocz jeśli większy lub równy (ze znakiem)

BRGE k

Operation:

(i) If Rd ≥ Rr (N ⊕ V = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRGE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k100

Skocz jeśli mniejszy niż (ze znakiem)

BRLT k

Operation:

(i) If Rd < Rr (N ⊕ V = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRLT k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k100

Skocz jeśli Half Carry jest ustawiony

BRHS k

Operation:

(i) If H = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	BRHS k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k101

Skocz jeśli Half Carry jest wyzerowany

BRHC k

Operation:

(i) If H = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	BRHC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k101

Skocz jeśli bit T wyzerowany

BRTC k

Operation:

(i) If T = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRTC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k110

Skocz jeśli bit T Ustawiony

BRTS k

Operation:

(i) If T = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	BRTS k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k110

Skocz jeśli bit przepełnienia ustawiony

BRVS k

Operation:

(i) If V = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRVS k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k011

Skocz jeśli bit przepełnienia wyzerowany

BRVC k

Operation:

(i) If V = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRVC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k011

Skocz jeśli bit maski przerwań ustawiony

BRIE k

Operation:

(i) If I = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRIE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k111

Skocz jeśli bit maski przerwań wyzerowany

BRID k

Operation:

(i) If I = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	BRID k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k111

Instrukcje przesyłu danych

Kopiowanie rejestrów

MOV Rd,Rr

Operation:

(i) Rd ← Rr

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0010	11rd	dddd	rrrr

Kopiowanie par rejestrów

MOVW Rd,Rr

Operation:

(i) Rd+1:Rd ← Rr+1:Rr

	Syntax:	Operands:	Program Counter:
(i)	MOVW Rd+1:Rd,Rr+1:Rr	Rd ∈ {0,2,...,30}, r ∈ {0,2,...,30}	PC ← PC + 1

16-bit Opcode:

0000	0001	dddd	rrrr

Ładowanie danej natychmiastowej do rejestru

LDI Rd,K

Operation:

(i) Rd ← K

	Syntax:	Operands:	Program Counter:
(i)	LDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

1110	KKKK	dddd	KKKK

Ładowanie pośrednie danej z pamięci do rejestru

LDS Rd,k

Operation:

(i) Rd ← (k)

	Syntax:	Operands:	Program Counter:
(i)	LDS Rd,k	0 ≤ d ≤ 31, 0 ≤ k ≤ 65535	PC ← PC + 2

32-bit Opcode:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Ładowanie pośrednie danej z pamięci do rejestru (16 bitów)

LDS (16bit) Rd,k

Operation:

(i) Rd ← (k)

	Syntax:	Operands:	Program Counter:
(i)	LDS Rd,k	16 ≤ d ≤ 31, 0 ≤ k ≤ 127	PC ← PC + 1

16-bit Opcode:

1010	0kkk	dddd	kkkk

Ładowanie pośrednie danej do rejestru

LD Rd,X

Using the X-pointer:

	Operation:		Comment:
(i)	Rd ← (X)		X: Unchanged
(ii)	Rd ← (X)	X ← X + 1	X: Post incremented
(iii)	X ← X - 1	Rd ← (X)	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, X	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, X+	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LD Rd, -X	0 ≤ d ≤ 31	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z postinkrementacją

LD Rd,X+

Using the X-pointer:

	Operation:		Comment:
(i)	Rd ← (X)		X: Unchanged
(ii)	Rd ← (X)	X ← X + 1	X: Post incremented
(iii)	X ← X - 1	Rd ← (X)	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, X	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, X+	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LD Rd, -X	0 ≤ d ≤ 31	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z predekrementacją

LD Rd,-X

Using the X-pointer:

	Operation:		Comment:
(i)	Rd ← (X)		X: Unchanged
(ii)	Rd ← (X)	X ← X + 1	X: Post incremented
(iii)	X ← X - 1	Rd ← (X)	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, X	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, X+	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LD Rd, -X	0 ≤ d ≤ 31	PC ← PC + 1

Ładowanie pośrednie danej do rejestru

LD Rd,Y

Using the Y-pointer:

	Operation:		Comment:
(i)	Rd ← (Y)		Y: Unchanged
(ii)	Rd ← (Y)	Y ← Y + 1	Y: Post incremented
(iii)	Y ← Y - 1	Rd ← (Y)	Y: Pre decremented
(iv)	Rd ← (Y+q)		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Y	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Y+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Y	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Y+q	$0 \le d \le 31, 0 \le q \le 63$	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z postinkrementacją

LD Rd,Y+

Using the Y-pointer:

	Operation:		Comment:
(i)	Rd ← (Y)		Y: Unchanged
(ii)	Rd ← (Y)	Y ← Y + 1	Y: Post incremented
(iii)	Y ← Y - 1	Rd ← (Y)	Y: Pre decremented
(iv)	Rd ← (Y+q)		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Y	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Y+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Y	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Y+q	$0 \le d \le 31$, $0 \le q \le 63$	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z predekrementacją

LD Rd,-Y

Using the Y-pointer:

	Operation:		Comment:
(i)	Rd ← (Y)		Y: Unchanged
(ii)	Rd ← (Y)	Y ← Y + 1	Y: Post incremented
(iii)	Y ← Y - 1	Rd ← (Y)	Y: Pre decremented
(iv)	Rd ← (Y+q)		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Y	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Y+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Y	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Y+q	$0 \le d \le 31, 0 \le q \le 63$	PC ← PC + 1

Ładowanie pośrednie danej do rejestru

LD Rd,Z

Using the Z-pointer:

	Operation:	Comment:	
(i)	Rd ← (Z)		Z: Unchanged
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Post increment
(iii)	Z ← Z -1	Rd ← (Z)	Z: Pre decrement
(iv)	Rd ← (Z+q)		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LD Rd, -Z	0 ≤ d ≤ 31	PC ← PC + 1
(iv)	LDD Rd, Z+q	0 ≤ d ≤ 31, 0 ≤ q ≤ 63	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z postinkrementacją

LD Rd,Z+

Using the Z-pointer:

	Operation:	Comment:	
(i)	Rd ← (Z)		Z: Unchanged
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Post increment
(iii)	Z ← Z -1	Rd ← (Z)	Z: Pre decrement
(iv)	Rd ← (Z+q)		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LD Rd, -Z	0 ≤ d ≤ 31	PC ← PC + 1
(iv)	LDD Rd, Z+q	0 ≤ d ≤ 31, 0 ≤ q ≤ 63	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z predekrementacją

LD Rd,-Z

Using the Z-pointer:

	Operation:	Comment:	
(i)	Rd ← (Z)		Z: Unchanged
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Post increment
(iii)	Z ← Z -1	Rd ← (Z)	Z: Pre decrement
(iv)	Rd ← (Z+q)		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(ii)	LD Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LD Rd, -Z	0 ≤ d ≤ 31	PC ← PC + 1
(iv)	LDD Rd, Z+q	0 ≤ d ≤ 31, 0 ≤ q ≤ 63	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z przesunięciem

LD Rd,Y+q

Using the Z-pointer:

	Operation:	Comment:	
(i)	Rd ← (Z)		Z: Unchanged
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Post increment
(iii)	Z ← Z -1	Rd ← (Z)	Z: Pre decrement
(iv)	Rd ← (Z+q)		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Z+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Z	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Z+q	$0 \le d \le 31$, $0 \le q \le 63$	PC ← PC + 1

Ładowanie pośrednie danej do rejestru z przesunięciem

LD Rd,Z+q

Using the Z-pointer:

	Operation:	Comment:	
(i)	Rd ← (Z)		Z: Unchanged
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Post increment
(iii)	Z ← Z -1	Rd ← (Z)	Z: Pre decrement
(iv)	Rd ← (Z+q)		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Z+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd, -Z	$0 \le d \le 31$	PC ← PC + 1
(iv)	LDD Rd, Z+q	$0 \le d \le 31$, $0 \le q \le 63$	PC ← PC + 1

Wysłanie danej z rejestru do komórki pamięci o adresie bezpośrednim

STS k,Rr

Operation:

(i) (k) ← Rr

	Syntax:	**Operands:**	**Program Counter:**
(i)	STS k,Rr	$0 \le r \le 31, 0 \le k \le 65535$	PC ← PC + 2

32-bit Opcode:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Wysłanie danej z rejestru do komórki pamięci o adresie bezpośrednim (16 bitów)

STS k,Rr

Operation:

(i) (k) ← Rr

	Syntax:	Operands:	Program Counter:
(i)	STS k,Rr	16 ≤ r ≤ 31, 0 ≤ k ≤ 127	PC ← PC + 1

16-bit Opcode:

1010	1kkk	dddd	kkkk

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w X

ST X,Rr

	Operation:		Comment:
(i)	(X) ← Rr		X: Unchanged
(ii)	(X) ← Rr	X ← X+1	X: Post incremented
(iii)	X ← X - 1	(X) ← Rr	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	ST X, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST X+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -X, Rr	$0 \le r \le 31$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w X z postinkrementacją

ST X+,Rr

	Operation:		Comment:
(i)	(X) ← Rr		X: Unchanged
(ii)	(X) ← Rr	X ← X+1	X: Post incremented
(iii)	X ← X - 1	(X) ← Rr	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	ST X, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST X+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -X, Rr	$0 \le r \le 31$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w X z predekrementacją

ST -X,Rr

	Operation:		Comment:
(i)	(X) ← Rr		X: Unchanged
(ii)	(X) ← Rr	X ← X+1	X: Post incremented
(iii)	X ← X - 1	(X) ← Rr	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	ST X, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST X+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -X, Rr	$0 \le r \le 31$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Y

ST Y,Rr

	Operation:		Comment:
(i)	(Y) ← Rr		Y: Unchanged
(ii)	(Y) ← Rr	Y ← Y+1	Y: Post incremented
(iii)	Y ← Y - 1	(Y) ← Rr	Y: Pre decremented
(iv)	(Y+q) ← Rr		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Y+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Y+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Y z postinkrementacją

ST Y+,Rr

	Operation:		Comment:
(i)	(Y) ← Rr		Y: Unchanged
(ii)	(Y) ← Rr	Y ← Y+1	Y: Post incremented
(iii)	Y ← Y - 1	(Y) ← Rr	Y: Pre decremented
(iv)	(Y+q) ← Rr		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Y+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Y+q, Rr	$0 \le r \le 31$, $0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Y z predekrementacją

ST -Y,Rr

	Operation:		Comment:
(i)	(Y) ← Rr		Y: Unchanged
(ii)	(Y) ← Rr	Y ← Y+1	Y: Post incremented
(iii)	Y ← Y - 1	(Y) ← Rr	Y: Pre decremented
(iv)	(Y+q) ← Rr		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Y+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Y+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Y z przsunięciem

STD Y+q,Rr

	Operation:		Comment:
(i)	(Y) ← Rr		Y: Unchanged
(ii)	(Y) ← Rr	Y ← Y+1	Y: Post incremented
(iii)	Y ← Y - 1	(Y) ← Rr	Y: Pre decremented
(iv)	(Y+q) ← Rr		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Y+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Y+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Z

ST Z,Rr

	Operation:		**Comment:**
(i)	(Z) ←Rr		Z: Unchanged
(ii)	(Z) ← Rr	Z ← Z+1	Z: Post incremented
(iii)	Z ← Z - 1	(Z) ← Rr	Z: Pre decremented
(iv)	(Z+q) ← Rr		Z: Unchanged, q: Displacement

	Syntax:	**Operands:**	**Program Counter:**
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Z z postinkrementacją

ST Z+,Rr

	Operation:		Comment:
(i)	(Z) ←Rr		Z: Unchanged
(ii)	(Z) ← Rr	Z ← Z+1	Z: Post incremented
(iii)	Z ← Z - 1	(Z) ← Rr	Z: Pre decremented
(iv)	(Z+q) ← Rr		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Z z predekrementacją

ST -Z,Rr

	Operation:		Comment:
(i)	(Z) ←Rr		Z: Unchanged
(ii)	(Z) ← Rr	Z ← Z+1	Z: Post incremented
(iii)	Z ← Z - 1	(Z) ← Rr	Z: Pre decremented
(iv)	(Z+q) ← Rr		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Przesłanie danej z rejestru do komórki pamięci o adresie zawartym w Z z przsunięciem

STD Z+q,Rr

	Operation:		Comment:
(i)	(Z) ←Rr		Z: Unchanged
(ii)	(Z) ← Rr	Z ← Z+1	Z: Post incremented
(iii)	Z ← Z - 1	(Z) ← Rr	Z: Pre decremented
(iv)	(Z+q) ← Rr		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

Załaduj daną z pamięci programu (spod adresu zawartego w Z) do rejestru R0

LPM

	Operation:		Comment:
(i)	R0 ← (Z)		Z: Unchanged, R0 implied destination register
(ii)	Rd ← (Z)		Z: Unchanged
(iii)	Rd ← (Z)	Z ← Z + 1	Z: Post incremented

	Syntax:	Operands:	Program Counter:
(i)	LPM	None, R0 implied	PC ← PC + 1
(ii)	LPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

Załaduj daną z pamięci programu (spod adresu zawartego w Z) do rejestru Rd

LPM Rd,Z

	Operation:		Comment:
(i)	R0 ← (Z)		Z: Unchanged, R0 implied destination register
(ii)	Rd ← (Z)		Z: Unchanged
(iii)	Rd ← (Z)	Z ← Z + 1	Z: Post incremented

	Syntax:	Operands:	Program Counter:
(i)	LPM	None, R0 implied	PC ← PC + 1
(ii)	LPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

Załaduj daną z pamięci programu (spod adresu zawartego w Z) do rejestru Rd z postinkrementacją

LPM Rd,Z+

	Operation:		Comment:
(i)	R0 ← (Z)		Z: Unchanged, R0 implied destination register
(ii)	Rd ← (Z)		Z: Unchanged
(iii)	Rd ← (Z)	Z ← Z + 1	Z: Post incremented

	Syntax:	Operands:	Program Counter:
(i)	LPM	None, R0 implied	PC ← PC + 1
(ii)	LPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	LPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

Załaduj daną z pamięci programu (spod adresu zawartego w Z i RAMPZ) do rejestru R0

ELPM

	Operation:		Comment:
(i)	R0 ← (RAMPZ:Z)		RAMPZ:Z: Unchanged, R0 implied destination register
(ii)	Rd ← (RAMPZ:Z)		RAMPZ:Z: Unchanged
(iii)	Rd ← (RAMPZ:Z)	(RAMPZ:Z) ← (RAMPZ:Z) + 1	RAMPZ:Z: Post incremented

	Syntax:	Operands:	Program Counter:
(i)	ELPM	None, R0 implied	PC ← PC + 1
(ii)	ELPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	ELPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

Załaduj daną z pamięci programu (spod adresu zawartego w Z i RAMPZ) do rejestru Rd

ELPM Rd,Z

	Operation:		Comment:
(i)	R0 ← (RAMPZ:Z)		RAMPZ:Z: Unchanged, R0 implied destination register
(ii)	Rd ← (RAMPZ:Z)		RAMPZ:Z: Unchanged
(iii)	Rd ← (RAMPZ:Z)	(RAMPZ:Z) ← (RAMPZ:Z) + 1	RAMPZ:Z: Post incremented

	Syntax:	Operands:	Program Counter:
(i)	ELPM	None, R0 implied	PC ← PC + 1
(ii)	ELPM Rd, Z	$0 \le d \le 31$	PC ← PC + 1
(iii)	ELPM Rd, Z+	$0 \le d \le 31$	PC ← PC + 1

Załaduj daną z pamięci programu (spod adresu zawartego w Z i RAMPZ) do rejestru Rd z postinkrementacją

ELPM Rd,Z+

	Operation:		Comment:
(i)	R0 ← (RAMPZ:Z)		RAMPZ:Z: Unchanged, R0 implied destination register
(ii)	Rd ← (RAMPZ:Z)		RAMPZ:Z: Unchanged
(iii)	Rd ← (RAMPZ:Z)	(RAMPZ:Z) ← (RAMPZ:Z) + 1	RAMPZ:Z: Post incremented

	Syntax:	Operands:	Program Counter:
(i)	ELPM	None, R0 implied	PC ← PC + 1
(ii)	ELPM Rd, Z	0 ≤ d ≤ 31	PC ← PC + 1
(iii)	ELPM Rd, Z+	0 ≤ d ≤ 31	PC ← PC + 1

Wyślij daną 16-to bitową z rejestrów R0,R1 do komórki pamięci programu (o adresie zawartym w Z + RAMPZ)

SPM

	Operation:	Comment:
(i)	(RAMPZ:Z) ← $ffff	Erase Program memory page
(ii)	(RAMPZ:Z) ← R1:R0	Write Program memory word
(iii)	(RAMPZ:Z) ← R1:R0	Write temporary page buffer
(iv)	(RAMPZ:Z) ← TEMP	Write temporary page buffer to Program memory
(v)	BLBITS ← R1:R0	Set Boot Loader Lock bits

	Syntax:	Operands:	Program Counter:
(i)-(v)	SPM	Z+	PC ← PC + 1

16-bit Opcode:

1001	0101	1110	1000

Wyślij daną 16-to bitową z rejestrów R0,R1 do komórki pamięci programu (o adresie zawartym w Z + RAMPZ) z postinkrementacją

SPM Z+

	Operation:	Comment:
(i)	(RAMPZ:Z) ← $ffff	Erase Program memory page
(ii)	(RAMPZ:Z) ← R1:R0	Write Program memory word
(iii)	(RAMPZ:Z) ← R1:R0	Write temporary page buffer
(iv)	(RAMPZ:Z) ← TEMP	Write temporary page buffer to Program memory
(v)	BLBITS ← R1:R0	Set Boot Loader Lock bits

	Syntax:	Operands:	Program Counter:
(i)-(v)	SPM	Z+	PC ← PC + 1

16-bit Opcode:

1001	0101	1110	1000

Ładowanie danej z urządzenia we-wy o adresie A do rejestru Rd

IN Rd,A

Operation:

(i) Rd ← I/O(A)

	Syntax:	Operands:	Program Counter:
(i)	IN Rd,A	$0 \le d \le 31, 0 \le A \le 63$	PC ← PC + 1

16-bit Opcode:

1011	0AAd	dddd	AAAA

Załaduj daną z rejestru Rr do urządzenia we-wy o adresie A

OUT A,Rr

Operation:

(i) I/O(A) ← Rr

	Syntax:	Operands:	Program Counter:
(i)	OUT A,Rr	$0 \leq r \leq 31, 0 \leq A \leq 63$	PC ← PC + 1

16-bit Opcode:

1011	1AAr	rrrr	AAAA

Składanie na stosie zawartości rejestru Rr

PUSH Rr

Operation:

(i) STACK ← Rr

	Syntax:	Operands:	Program Counter:	Stack:
(i)	PUSH Rr	0 ≤ r ≤ 31	PC ← PC + 1	SP ← SP - 1

16-bit Opcode:

1001	001d	dddd	1111

Ściągnięcie ze stosu danej do rejestru Rd

POP Rd

Operation:

(i) Rd ← STACK

	Syntax:	Operands:	Program Counter:	Stack:
(i)	POP Rd	0 ≤ d ≤ 31	PC ← PC + 1	SP ← SP + 1

16-bit Opcode:

1001	000d	dddd	1111

Wymiana zawartości rejestru Rd z zawartością komórki pamięci o adresie zawartym w Z

XCH Z,Rd

Operation:

(i) (Z) ← Rd, Rd ← (Z)

	Syntax:	Operands:	Program Counter:
(i)	XCH Z,Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	001r	rrrr	0100

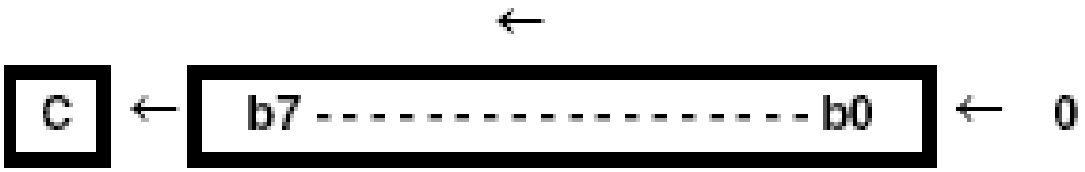
Operacje na bitach i testowanie bitów

Przesunięcie logiczne w lewo

LSL Rd

Operation:

(i)



	Syntax:	Operands:	Program Counter:
(i)	LSL Rd	$0 \le d \le 31$	PC ← PC + 1

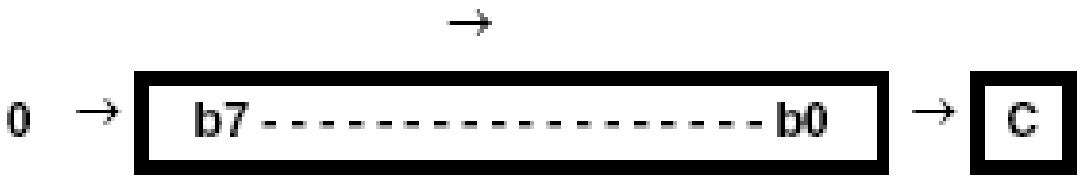
16-bit Opcode: (see ADD Rd,Rd)

0000	11dd	dddd	dddd

Przesunięcie logiczne w prawo

LSR Rd

Operation:



	Syntax:	Operands:	Program Counter:
(i)	LSR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

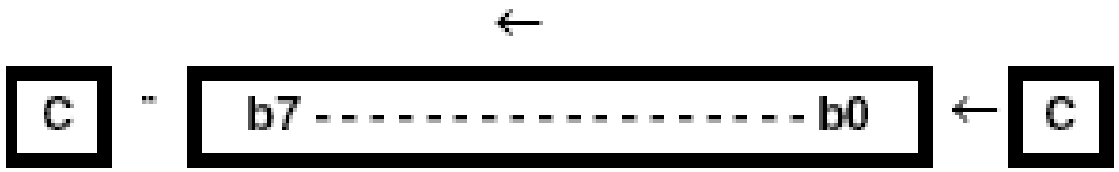
16-bit Opcode:

1001	010d	dddd	0110

Rotacja w lewo z Carry

ROL Rd

Operation:



	Syntax:	Operands:	Program Counter:
(i)	ROL Rd	$0 \le d \le 31$	PC ← PC + 1

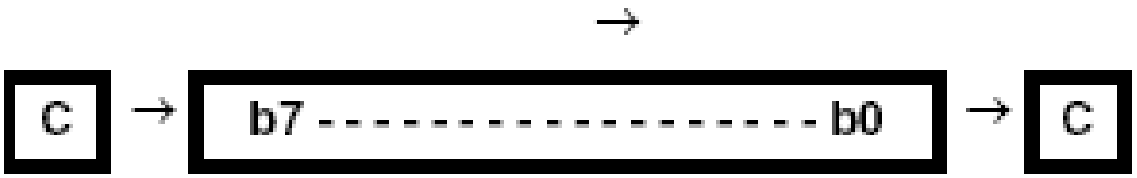
16-bit Opcode: (see ADC Rd,Rd)

0001	11dd	dddd	dddd

Rotacja w prawo z Carry

ROR Rd

Operation:



	Syntax:	Operands:	Program Counter:
(i)	ROR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

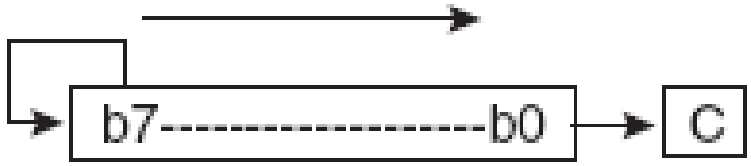
1001	010d	dddd	0111

Przesunięcie arytmetyczne w prawo

ASR Rd

Operation:

(i)



	Syntax:	Operands:	Program Counter:
(i)	ASR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	010d	dddd	0101

Zamiana czterech starszych bitów rejestru z czterema starszymi bitami

SWAP

Operation:

(i) R(7:4) ← Rd(3:0), R(3:0) ← Rd(7:4)

	Syntax:	Operands:	Program Counter:
(i)	SWAP Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0010

Ustaw bit w rejestrze statusowym SREG

BSET s

Operation:

(i) SREG(s) ← 1

	Syntax:	Operands:	Program Counter:
(i)	BSET s	0 ≤ s ≤ 7	PC ← PC + 1

16-bit Opcode:

1001	0100	0sss	1000

Zeruj bit w rejestrze statusowym SREG

BCLR s

Operation:

(i) SREG(s) ← 0

Syntax:	Operands:	Program Counter:
(i) BCLR s	0 ≤ s ≤ 7	PC ← PC + 1

16-bit Opcode:

1001	0100	1sss	1000

Ustawienie bitu b w rejestrze we-wy o adresie A

SBI A,b

Operation:

(i) I/O(A,b) ← 1

	Syntax:	Operands:	Program Counter:
(i)	SBI A,b	$0 \le A \le 31, 0 \le b \le 7$	PC ← PC + 1

16-bit Opcode:

1001	1010	AAAA	Abbb

Zeruj bit b w rejestrze we-wy o adresie A

CBI A,b

Operation:

(i) I/O(A,b) ← 0

	Syntax:	Operands:	Program Counter:
(i)	CBI A,b	0 ≤ A ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

16-bit Opcode:

1001	1000	AAAA	Abbb

Skopiuj bit b z rejestru Rr do bitu T

BST Rr,b

Operation:

(i) T ← Rd(b)

	Syntax:	Operands:	Program Counter:
(i)	BST Rd,b	0 ≤ d ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

16-bit Opcode:

1111	101d	dddd	0bbb

Skopiuj bit T do bitu b w rejestrze Rd

BLD Rd,b

Operation:

(i) Rd(b) ← T

	Syntax:	Operands:	Program Counter:
(i)	BLD Rd,b	0 ≤ d ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

16 bit Opcode:

1111	100d	dddd	0bbb

Ustawianie bitu Carry

SEC

Operation:

(i) C ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEC	None	PC ← PC + 1

16-bit Opcode:

1001	0100	0000	1000

Kasowanie bitu Carry

CLC

Operation:

(i) $C \leftarrow 0$

Syntax:	**Operands:**	**Program Counter:**
(i) CLC	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1000	1000

Ustawianie bitu N

SEN

Operation:

(i) N ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEN	None	PC ← PC + 1

16-bit Opcode:

1001	0100	0010	1000

Kasowanie bitu N

CLN

Operation:

(i) N ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLN	None	PC ← PC + 1

16-bit Opcode:

1001	0100	1010	1000

Ustawianie bitu zera

SEZ

Operation:

(i) $Z \leftarrow 1$

	Syntax:	Operands:	Program Counter:
(i)	SEZ	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	0001	1000

Zerowanie bitu Z

CLZ

Operation:

(i) $Z \leftarrow 0$

	Syntax:	Operands:	Program Counter:
(i)	CLZ	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1001	1000

Ustawianie bitu globalnej maski przerwań

SEI

Operation:

(i) I ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEI	None	PC ← PC + 1

16-bit Opcode:

1001	0100	0111	1000

Kasowanie bitu globalnej maski przerwań

CLI

Operation:

(i) $I \leftarrow 0$

	Syntax:	Operands:	Program Counter:
(i)	CLI	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1111	1000

Ustawianie bitu S

SES

Operation:

(i) S ← 1

	Syntax:	Operands:	Program Counter:
(i)	SES	None	PC ← PC + 1

16-bit Opcode:

1001	0100	0100	1000

Kasowanie bitu S

CLS

Operation:

(i) S ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLS	None	PC ← PC + 1

16-bit Opcode:

1001	0100	1100	1000

Ustawianie bitu V

SEV

Operation:

(i) V ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEV	None	PC ← PC + 1

16-bit Opcode:

1001	0100	0011	1000

Zerowanie bitu V

CLV

Operation:

(i) $V \leftarrow 0$

	Syntax:	Operands:	Program Counter:
(i)	CLV	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1011	1000

Ustawianie bitu T

SET

Operation:

(i) $T \leftarrow 1$

	Syntax:	Operands:	Program Counter:
(i)	SET	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	0110	1000

Kasowanie bitu T

CLT

Operation:

(i) T ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLT	None	PC ← PC + 1

16-bit Opcode:

1001	0100	1110	1000

Ustawianie bitu H

SEH

Operation:

(i) H ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEH	None	PC ← PC + 1

16-bit Opcode:

1001	0100	0101	1000

Kasowanie bitu H

CLH

Operation:

(i) $H \leftarrow 0$

	Syntax:	Operands:	Program Counter:
(i)	CLH	None	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	1101	1000

Operacje sterujące

Normalnie nie używana w programie użytkownika służy do debagowania programu on-chip

BREAK

	Syntax:	Operands:	Program Counter:
(i)	BREAK	None	PC ← PC + 1

16-bit Opcode:

1001	0101	1001	1000

Operacja pusta

NOP

Operation:

(i) No

	Syntax:	**Operands:**	**Program Counter:**
(i)	NOP	None	PC ← PC + 1

16-bit Opcode:

0000	0000	0000	0000

Wprowadzenie do trybu uśpienia

SLEEP

Syntax:	Operands:	Program Counter:
SLEEP	None	PC ← PC + 1

16-bit Opcode:

1001	0101	1000	1000

Zerowanie Aatch Doga

WDR

Operation:

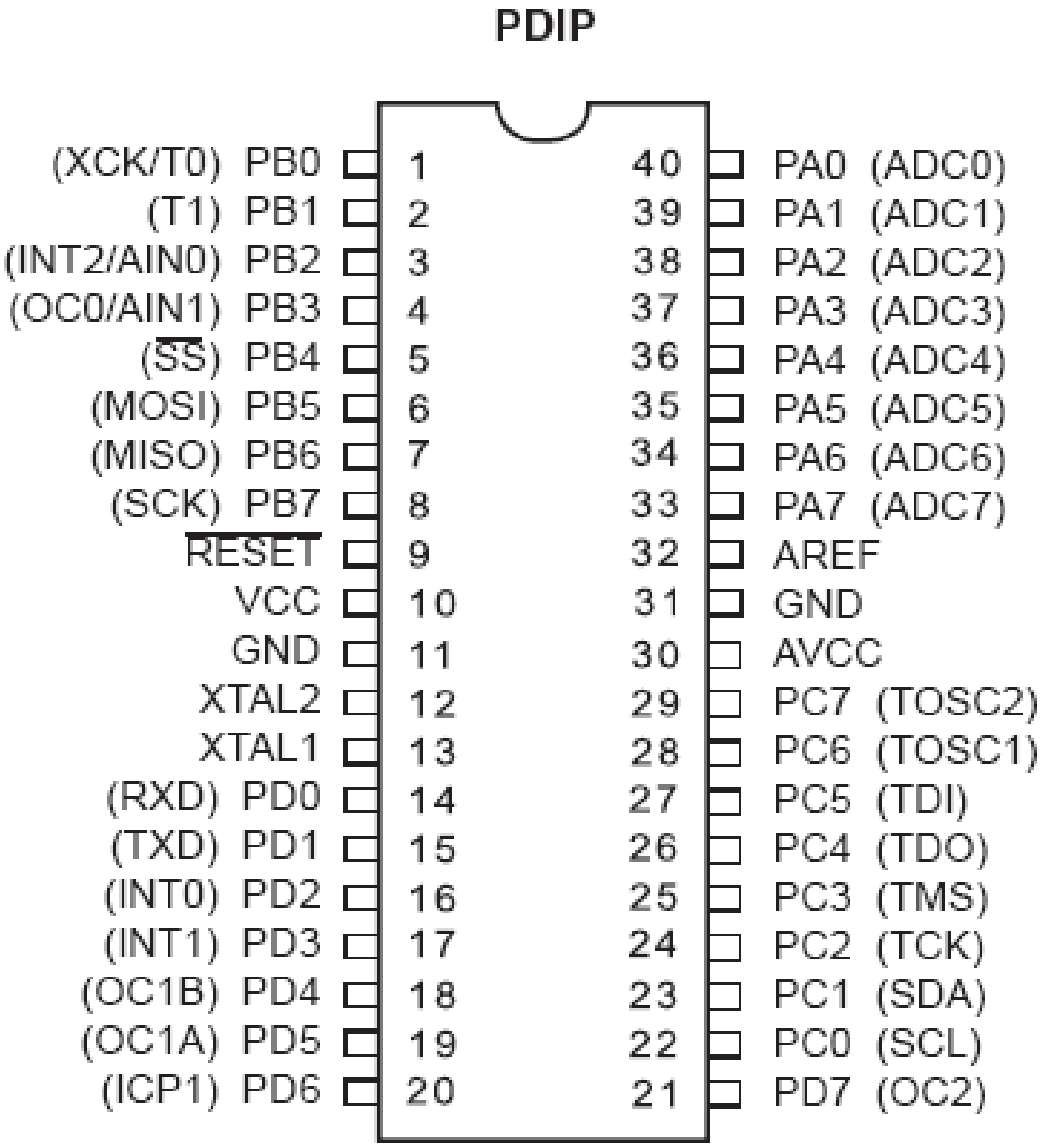
(i) WD timer restart.

Syntax:	**Operands:**	**Program Counter:**
(i) WDR	None	PC ← PC + 1

16-bit Opcode:

1001	0101	1010	1000

Wyprowadzenia układu ATmega32



VCC: zasilanie

GND: masa

PA0-PA7: port wejścia wyjścia z możliwością dołączania rezystorów podciągających lub wejścia przetwornika analogowo0cyfrowego

PB0-PB7: port wejścia wyjścia z możliwością dołączania rezystorów podciągających lub:

PB0: wejście licznika czasomierza T0 lub wejście taktowania portu szeregowego

PB1: wejście licznika czasomierza T1

PB2: Wejście dodatnie komparatora analogowego lub wejście przerywające INT2

PB3: Wejście ujemne komparatora analogowego lub wyście Output Compare licznika T0

PB4: SPI wejście Slave Select

PB5: SPI Master Output/Slave Input- MOSI

PB6: SPI Master Input/Slave Output- MISO

PB7: SPI sygnał zegarowy- SCK

PC0-PC7: port wejścia wyjścia z możliwością dołączania rezystorów podciągających lub:

PC0: Sygnał zegarowy magistrali I2C

PC1: Linia danych magistrali I2C

PC2: Zegar złącza JTAG

PC3: Złącze JTAG wybór modu

PC4: Wyjście danych złącza JTAG

PC5: Wejście danych złącza JTAG

PC6: Zewnętrzny oscylator licznika- TOSC1

PC7: Zewnętrzny oscylator licznika- TOSC2

PD0-PD7: port wejścia wyjścia z możliwością dołączania rezystorów podciągających lub:

PD0: RxD- wejście danych asynchronicznego portu szeregowego USART

PD1: TxD- wyjście danych asynchronicznego portu szeregowego USART

PD2: INT0- wejście przerywające 0

PD3: INT1- wejście przerywające 1

PD4: Wyjście A Output Compare licznika-czasomierza T1

PD5: Wyjście B Output Compare licznika-czasomierza T1

PD6: Wejście Input-Capture licznika czasomierza T1

PD7: Wyjście Output Compare licznika-czasomierza T2

RESET: wejście zerujące

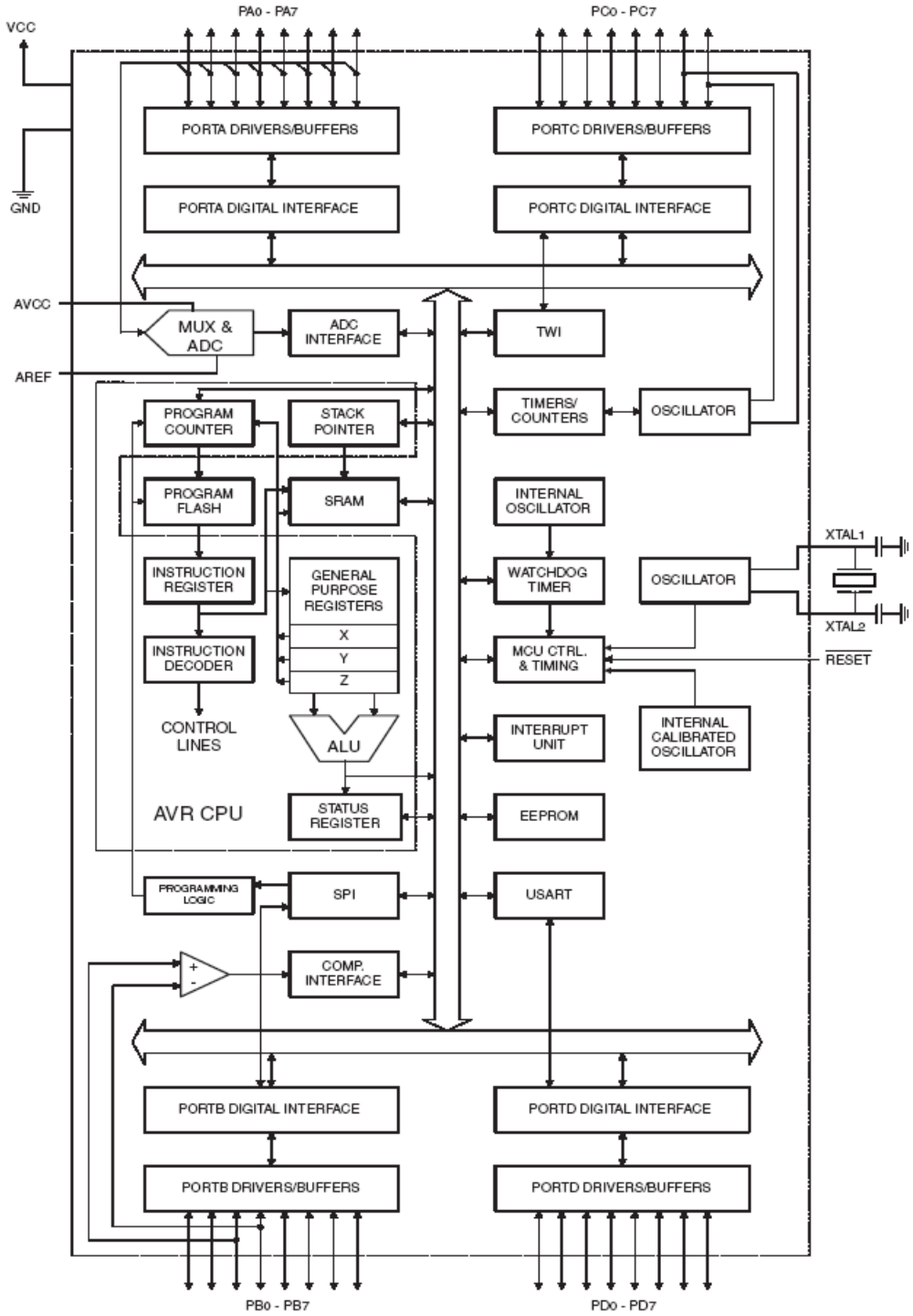
XTAL1: Zewnętrzny oscylator lub wejście zegara zewnętrznego

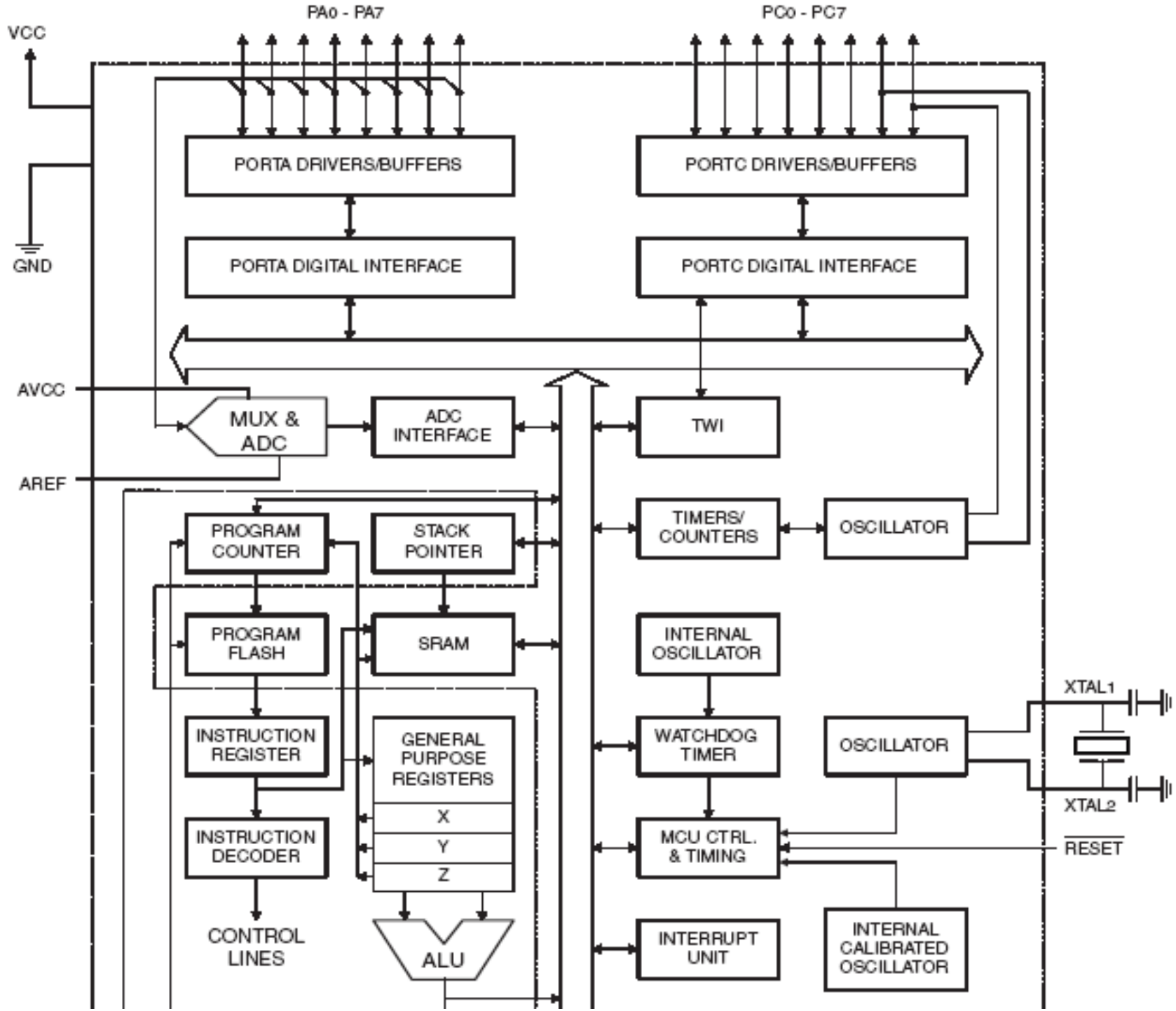
XTAL2: Zewnętrzny oscylator

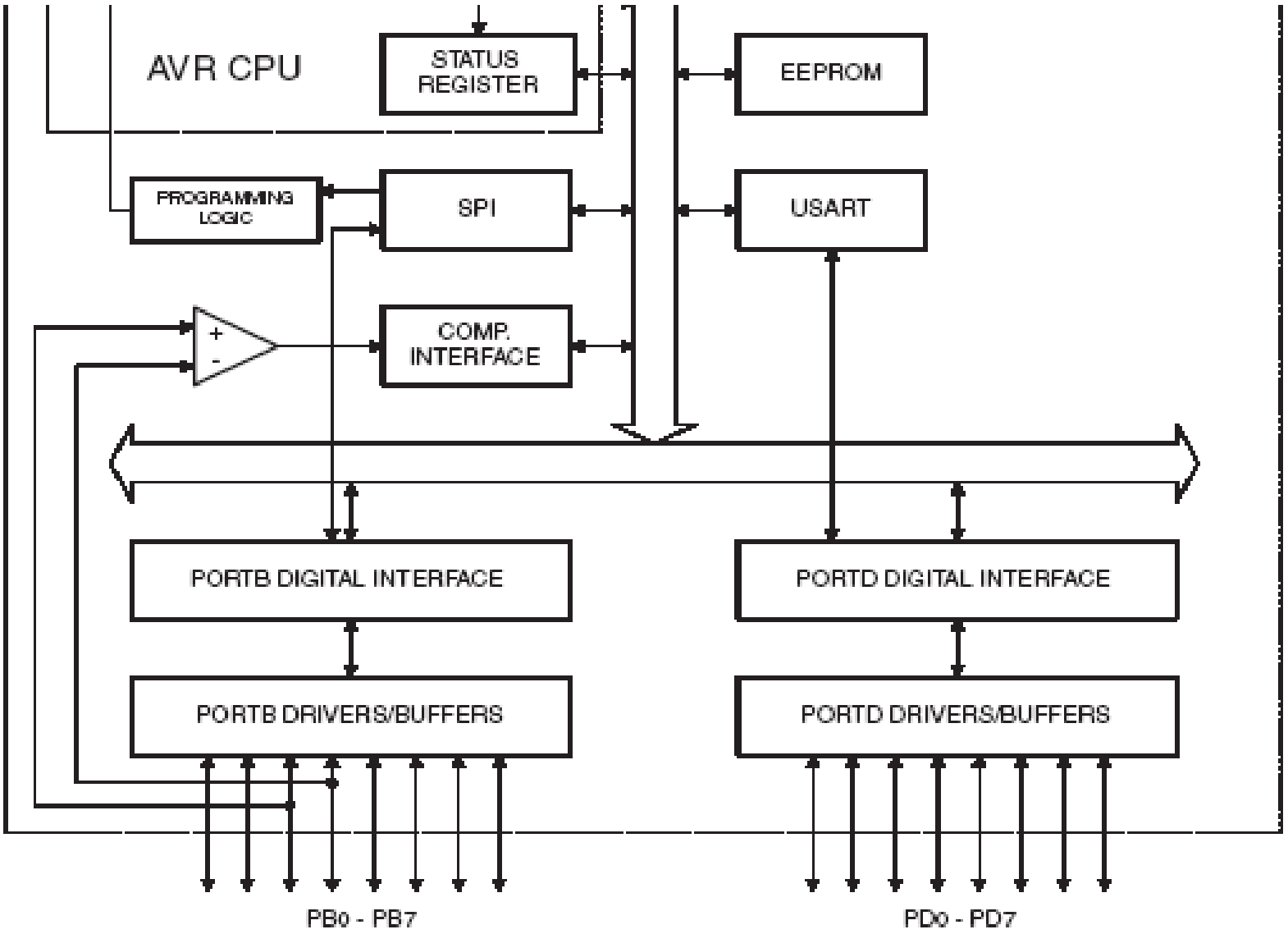
AVCC: wejście zasilania przetwornika AC (gdy nieużywany podłączony do VCC, gdy używany podłączony do VCC przez filtr dolonprzepustowy)

AREF: wejście napięcia referencyjnego dla przetwornika AC

Schemat blokowy




PA0 - PA7
PC0 - PC7
VCC
GND
PORTA DRIVERS/BUFFERS
PORTC DRIVERS/BUFFERS
PORTA DIGITAL INTERFACE
PORTC DIGITAL INTERFACE
AVCC
AREF
MUX & ADC
ADC INTERFACE
TWI
PROGRAM COUNTER
STACK POINTER
TIMERS/ COUNTERS
OSCILLATOR
PROGRAM FLASH
SRAM
INTERNAL OSCILLATOR
XTAL1
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTERS
X
Y
Z
WATCHDOG TIMER
OSCILLATOR
XTAL2
INSTRUCTION DECODER
MCU CTRL. & TIMING
RESET
CONTROL LINES
ALU
INTERRUPT UNIT
INTERNAL CALIBRATED OSCILLATOR


AVR CPU
STATUS REGISTER
EEPROM
PROGRAMMING LOGIC
SPI
USART
+
-
COMP. INTERFACE
PORTB DIGITAL INTERFACE
PORTD DIGITAL INTERFACE
PORTB DRIVERS/BUFFERS
PORTD DRIVERS/BUFFERS
PB0 - PB7
PD0 - PD7

Dostęp do pamięci EEPROM

Dostęp do pamięci EEPROM jest możliwy dzięki odpowiednim rejestrom lokowanym w przestrzeni wejścia-wyjścia.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

Rejestr adresowy EEPROM- EEARH i EEARL

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Rejestr danych EEPROM-EEDR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

Rejestr kontrolny pamięci EEPROM-EECR

BIT3: EERIE- generacja przerwania przy gotowości do zapisu, Gdy ustawiony gotowość pamięci jest równoznaczna ze zgłoszeniem dedykowanego przerwania. Jest ono generowane ciągle dopóki EEWE=„0”

BIT2: EEMWE- zezwolenie na zapis pamięci- musi być ustawiony na cztery cykle zegarowe przed zapisem pamięci EEPROM

BIT1: EEWE- zapis pamięci, jeśli EEMWE=„1” i pamięć nie jest zajęta zapisem poprzednim to ustawienie bitu EEWE powoduje zapis, a jego wyzerowanie informuje o końcu zapisu

BIT0: EERE- odczyt pamięci, jeśli pamięć nie jest zajęta ustawienie bitu powoduje odczyt zaadresowanej komórki do rejestru danych.

Źródła przerwań w Atmega32

nr	Adres obsł.	Źródło	Przyczyna
1	$000	Reset	Pin, Power on reset, Watch Dog, Brown out reset, JTAG reset
2	$002	INT0	Sygnał na końcówce INT0
3	$004	INT1	Sygnał na końcówce INT1
4	$006	INT2	Sygnał na końcówce INT2
5	$008	TIMER2 COMP	Licznik T2- funkcja porównania
6	$00A	TIMER2 OVF	Przepełnienie licznika T2
7	$00C	TIMER1 CAPT	Licznik T1- funkcja Input Capture
8	$00E	TIMER1 COMPA	Licznika T1- funkcja porównania A
9	$010	TIMER1 COMPB	Licznika T1- funkcja porównania B
10	$012	TIMER1 OVF	Przepełnienie licznika T1

Źródła przerwań w ATmega32

nr	Adres obsł.	Źródło	Przyczyna
11	$014	TIMER0 COMP	Licznik T0- funkcja porównania
12	$016	TIMER0 OVF	Przepełnienie licznika T0
13	$018	SPI STC	Transmisja szeregowa zakończona
14	$01A	USART RXC	Dana odebrana przez USART
15	$01C	USART UDRE	Rejestr danych USART pusty
16	$01E	USART TXC	Dana wysłana przez USART
17	$020	ADC	Koniec konwersji AC
18	$022	EE_RDY	Pamięć EEPROM gotowa
19	$024	ANA_COMP	Komparator analogowy
20	$026	TWI	I2C
21	$028	SPM_RDY	Zapis pamięci programu gotowy

Priorytet przerwania jest zgodny z jego numerem:

Im wyższy numer wektora przerwania tym niższy priorytet.