
Asembler
3

Elementy języka asembler

Elementy języka asembler

- Komentarze
- Etykiety
- Dyrektywy asemblera
- Instrukcje
- Operatory asemblera
- Funkcje asemblera

Elementy języka asembler

Elementy języka asembler

ÅKomentarze

; komentarz zaczyna się od średnika i jest informacją przydatną dla programisty

ÅEtykiety

Mogą poprzedzać każdą instrukcję lub dyrektywę. Umieszczenie etykiety przed instrukcją powoduje, że translator przypisuje jej nazwie wartość będącą adresem wskazywanej instrukcji. Etykiety przydatne są do określania miejsca skoków i rozgałęzień.

```
skok: nop
      nop
```

Elementy języka asembler

Elementy języka asembler

Dyrektywy

Dyrektywy w języku asembler sterują pracą kompilatora oraz definiują dodatkowe elementy programu. Dyrektywy poprzedzone są kropką.

.byte
.cseg
.db
.def
.device
.dseg
.dw
.macro, .endmacro
.equ
.eseg
.exit
.include
.list, .listmac, .nolist
.org
.set

Dalsza część zapisów dotyczy programu zawartego w pamięci FLASH. Miejsce pierwszego adresu programu definiujemy za pomocą dyrektywy .org. Brak dyrektywy .org powoduje automatyczne przyjęcie jako adresu początkowego wartości 0x000.

Dalsza część zapisów dotyczy definicji zmiennych w pamięci SRAM. Miejsce pierwszego adresu zmiennej definiujemy za pomocą dyrektywy .org. Brak dyrektywy .org automatycznie ustawia adres na wartość 0x0060 lub 0x0100.

Dalsza część zapisów dotyczy pamięci EEPROM. Miejsce pierwszego adresu zmiennej definiujemy za pomocą dyrektywy .org. Brak dyrektywy .org automatycznie ustawia adres na wartość 0x0000.

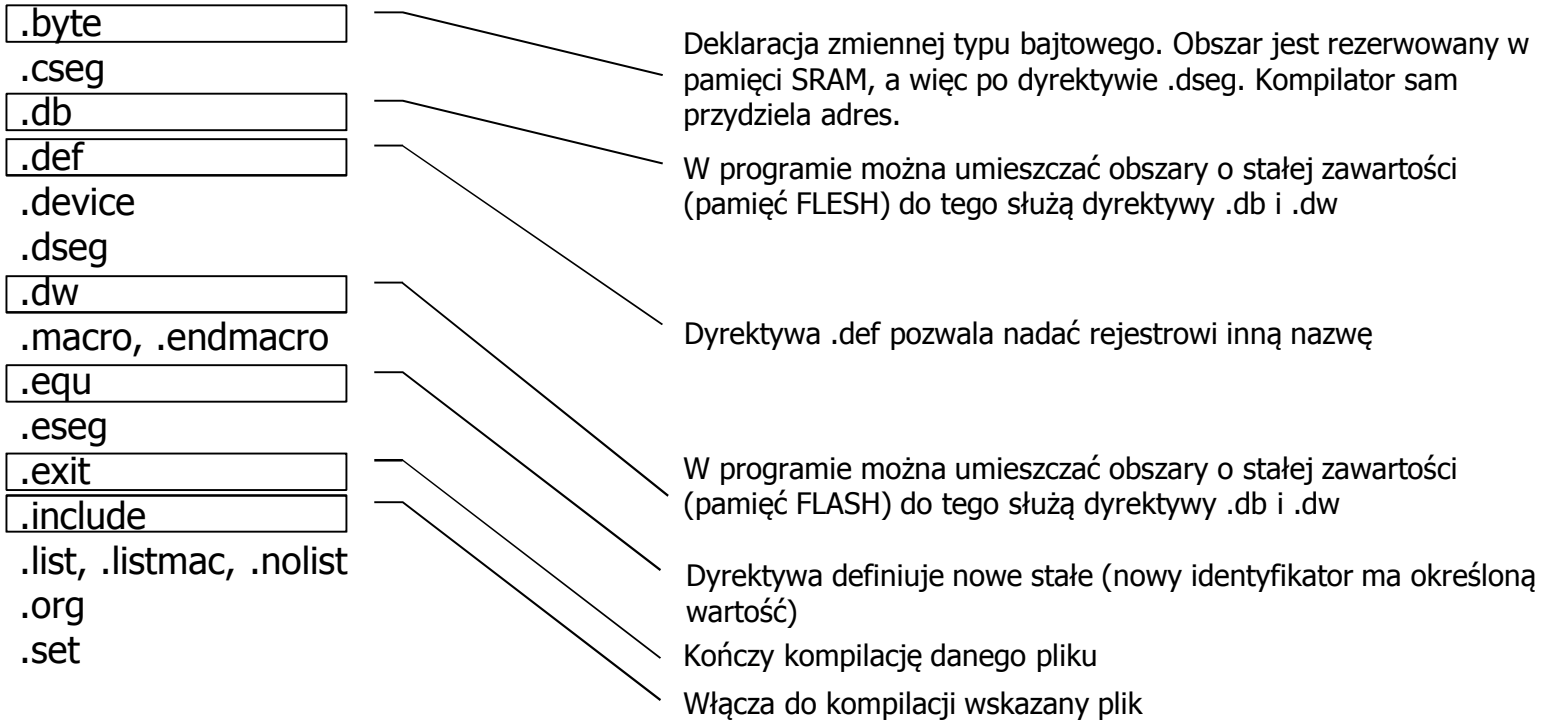
Dyrektywa .org określa adres gdzie kompilator zacznie umieszczanie generowanych bajtów

Elementy języka asembler

Elementy języka asembler

Dyrektywy

Dyrektywy w języku asembler sterują pracą kompilatora oraz definiują dodatkowe elementy programu. Dyrektywy poprzedzone są kropką.



Elementy języka asembler

Elementy języka asembler

Dyrektywy - Makra

Makra w języku asembler to struktury predefiniujące
bloki programowe o zmiennych parametrach. Wywołanie makra powoduje
zastąpienie nazwy makra jego rozwinięciem. Makra mogą mieć parametry.
Wykorzystujemy je za pomocą zapisu @numer-parametru.

.macro nazwa_makra

.endmacro

```
.macro dodawaj          ; definicja makra
add @0,@1
.endmacro


dodawaj R1,R2           ; wywołanie makra
```

Elementy języka asembler

Elementy języka asembler

Dyrektywy – zmienne w pamięci danych

Opis pamięci danych (fragment programu poprzedza dyrektywa .dseg)
wprowadzamy dyrektywą .byte

```
.dseg
.org            0x0060
x_1             .byte    1
Tablica         .byte    100
```

Dyrektywy – definicje

Dyrektywy umożliwiające przypisanie określonej wartości liczbowej
Do definiowanej etykiety

```
.equ        stala=9
.set        ala=9              ; etykiete można wielokrotnie zmieniac
```

Lista instrukcji mikrokontrolera

Grupy instrukcji:

- **Instrukcje arytmetyczne i logiczne**
- **Instrukcje skoków**
- **Instrukcje przesyłania danych**
- **Instrukcje operacji bitowych**
- **Instrukcje kontroli pracy procesora**

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Lista instrukcji mikrokontrolera

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd × Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd × Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd × Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd × Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd × Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd × Rr) << 1	Z,C	2

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Lista instrukcji mikrokontrolera

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

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Lista instrukcji mikrokontrolera

DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	Stack ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← Stack	None	2

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Lista instrukcji mikrokontrolera

BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1

Lista instrukcji mikrokontrolera

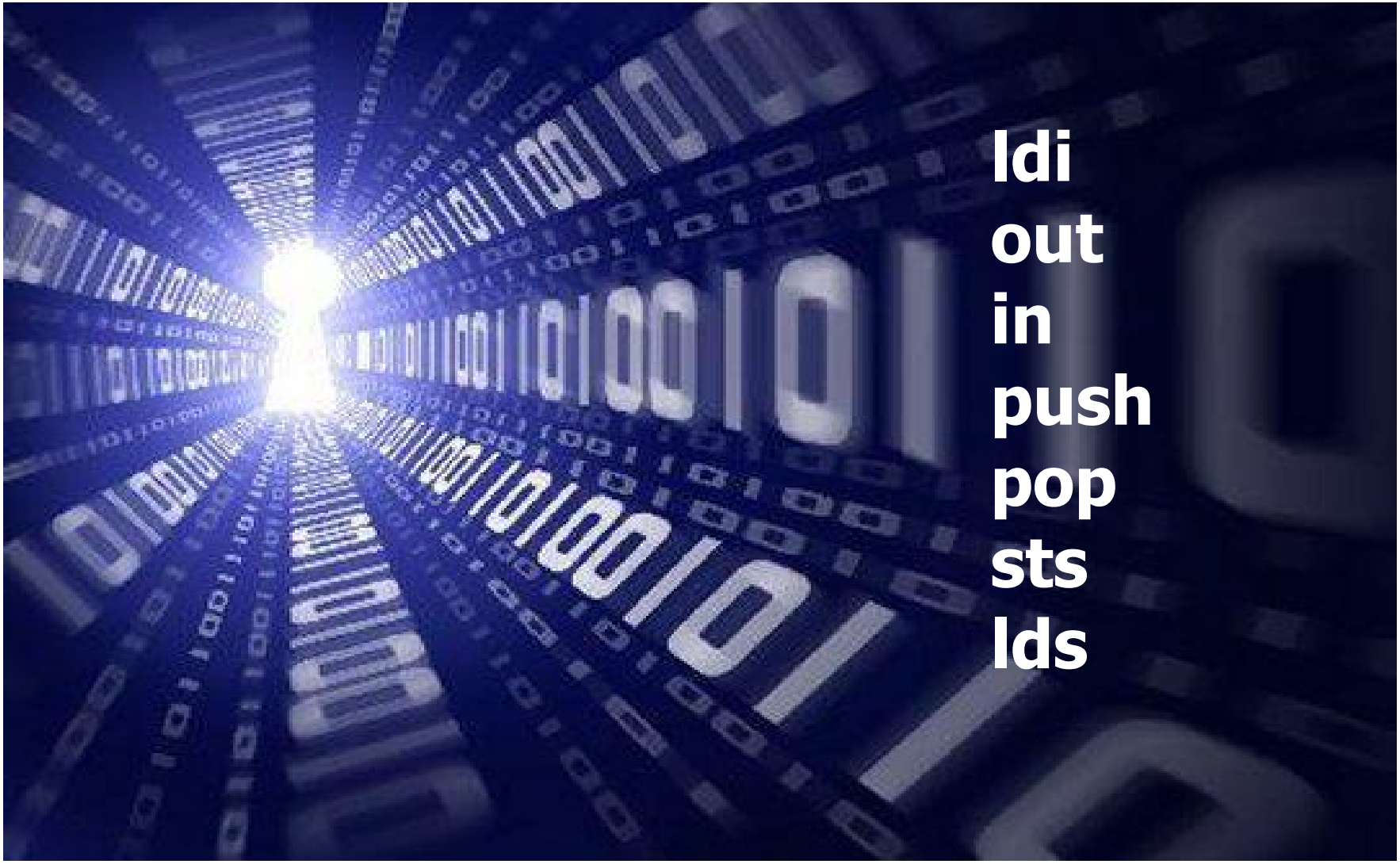
Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A

Symboliczne oznaczenia rejestrów i stałych

Symbol	Znaczenie
Rd	Rejestr przeznaczenia: R0 … R31
Rs	Rejestr źródła: R0 … R31
Rh	Górne rejestry: R16 … R31
Rhd	Górny rejestr przeznaczenia: R16 … R31
Rhs	Górny rejestr źródła: R16 … R31
Rm	Środkowy rejestr przeznaczenia: R16 … R23
Rxyz	Rejestry indeksowe: X=R27:R26, Y=R29:R28, R31:R30
adr4M	Stała 22 bitowa bez znaku, zakres 0…0x3ffff
adr64k	Stała 16 bitowa bez znaku, zakres 0…0xffff
adr2k	Stała 12 bitowa ze znakiem, zakres -2048…2047
adr64	Stała 7 bitowa ze znakiem, zakres -64…63

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Zapoznanie się z rozkazami



WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja ldi

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
ldi	**l**oa**d** **i**mmediate Bezpośrednie wpisanie do rejestru ośmiobitowej stałej	Rh ← c255 PC ← PC+1	- - - - - - - -

składnia
ldi Rh,c255
Uwaga tylko Rh !!!!!!!

Przykłady

```
.equ x =1

ldi r16,0x25
ldi r19,x
ldi r19,255
```

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja out

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
out	**Out**put port Zapisz wartość rejestru w przestrzeni we-wy	(P+0x20) ← Rs PC ← PC+1	- - - - - - - -

składnia
out P, Rs

Przykłady

out 0x3F,r1
out porta,r30

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja in

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
in	**In**put port Zapisz wartość z przestrzeni we-wy do rejestru Rs	Rd← (P+0x20) PC ← PC+1	- - - - - - - -

składnia
in Rd, P

Przykłady

in r1, 0x3F
in r23, porta

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja push

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
push	**Push** register on stack Prześlij zawartość rejestru Rs na wierzchołek stosu	(Sp) ←Rs Sp←SP-1 PC ← PC+1	- - - - - - - -

składnia
push Rs

Operuje tylko na rejestrach funkcyjnych

Przykłady

```
push r12
push r14

ldi r16,sreg
push r16
```

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja pop

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
pop	**POP** register from stack Prześlij wartość z wierzchołka stosu do rejestru Rd	Rd←(Sp) Sp=Sp+1 PC ← PC+1	- - - - - - - -

składnia
pop Rd

Przykłady

pop r12
pop r19

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja sts

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
sts	**St**ore direct to data space Zapisz zawartość rejestru w pamięci danych	(adr64k)←Rs PC ← PC+2	- - - - - - - -

składnia
sts adr64k,Rs

Operuje na całym obszarze danych
włączając rejestry robocze i funkcyjne

Przykłady

sts 100,r1 - zapisz zawartość rejestru do 100 komórki pamięci lub komórki pamięci o adresie 0x064

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja lds

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
lds	**Load** direct from data space Odczytaj wartość pamięci danych do rejestru	Rd←(adr64k) PC ← PC+2	- - - - - - - -

składnia
lds Rd,adr64k

Operuje na całym obszarze danych
włączając rejestry robocze i funkcyjne

Przykłady

lds r1,0x001 – zapisz zawartość komórki pamięci o adresie 0x001 do rejestru r1

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja jmp

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
jmp	**JuMP** Skok adresowany bezpośrednio	PC ← PC+adr4M	- - - - - - - -

składnia
jmp adr4M

Stała 22 bitowa bez znaku, zakres 0…0x3ffff 262143

Przykłady

```
main:   nop
        ldi r16,0x25
        jmp main
```

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Instrukcja rjmp

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
rjmp	**Relative JuMP** Skok adresowany względnie	PC ← PC+adr2k+1	- - - - - - - -

składnia
rjmp adr2k

Stała 12 bitowa ze znakiem, zakres -2048…2047

Przykłady

```
main:
        nop
        ldi r16,0x25
        rjmp main
```

Instrukcja rjmp działa najszybciej, dlatego zalecane jest jej stosowanie wszędzie tam, gdzie jest to możliwe (gdy długość skoku jest mniejsza lub równa ±2 ksłowa -2048…2047).