

POLITECHNIKA POZNAŃSKA

INSTYTUT AUTOMATYKI, ROBOTYKI I INŻYNIERII INFORMATYCZNEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ



PROGRAMOWA ORAZ SPRZĘTOWA IMPLEMENTACJA OPERACJI
LOGICZNYCH

TECHNIKA CYFROWA I MIKROPROCESOROWA

MATERIAŁY DO ZAJĘĆ LABORATORYJNYCH

DR INŻ. DOMINIK ŁUCZAK

DOMINIK.LUCZAK@PUT.POZNAN.PL

I. CEL

WIEDZY

Celem zajęć jest zapoznanie z:

- algebrą Boole'a,
- implementacją operacji logicznych za pomocą bramek logicznych oraz języka wysokiego poziomu,
- budową i działaniem cyfrowych wejść i wyjść mikrokontrolera,
- operatorami bitowymi dostępnymi w języku C,
- operatorami logicznymi dostępnymi w języku C,
- systemem obsługi przerwań mikrokontrolera.

UMIEJĘTNOŚCI

Celem zajęć jest nabycie umiejętności programistycznych pozwalających na programową obsługę cyfrowych wejść i wyjść mikrokontrolera:

- implementacją operacji logicznych za pomocą bramek logicznych oraz języka wysokiego poziomu,
- konfiguracja pojedynczej linii mikrokontrolera jako wejście lub wyjście cyfrowe,
- zmiana wartości binarnej pojedynczego wyjścia cyfrowego mikrokontrolera,
- odczyt wartości binarnej pojedynczego wejścia cyfrowego mikrokontrolera,
- wykorzystaniem makroinstrukcji do wykonania zapisu lub odczytu pojedynczej wartości binarnej,
- prosty program wykorzystujący jedno wejście i jedno wyjście cyfrowe,
- złożony program wykorzystujący wiele wejść i wiele wyjść cyfrowych,
- wykorzystanie przerwania od zdarzenia zewnętrznego na wejściu cyfrowym.

KOMPETENCJI SPOŁECZNYCH

Celem zajęć jest kształtowanie właściwych postaw programistycznych:

- tworzenie kodu programu w sposób czytelny,
- stosowanie dyrektyw preprocesora oraz makroinstrukcji,
- prawidłowa obsługa programowa przerwań mikrokontrolera,
- stosowanie stosownych komentarzy,
- weryfikacja poprawności działania programu.

II. POLECENIA KOŃCOWE

Dokończ zadania nie zrealizowane w trakcie zajęć. Przygotuj raport z przeprowadzonych zajęć. Jeden raport może być przygotowany przez maksymalnie 2 osoby. W projekcie zamieść:

1. Opis rozwiązania problemu oraz jego implementację.
2. Rysunki lub fotografie ilustrujące poprawne działanie programu.
3. Opis działania własnych programów. Udokumentuj przeprowadzenie testów funkcjonalnych z wykorzystaniem symulatora.
4. Ważne instrukcje programu należy zaopatrzyć stosownym komentarzem. Wszystkie przygotowane funkcje oraz zmienne globalne należy zaopatrzyć komentarzem zgodnym z składnią generatora dokumentacji (*Doxygen*).

Jako załącznik dołącz spakowane kody programów. Dla każdego zadania przygotuj osobny projekt. Raport należy przygotować i przekazać do kolejnego spotkania. Raporty przekazywane później nie będą oceniane.

Na ocenę raportu będą miały wpływ następujące elementy (łącznie 5 punktów):

1. spełnienie wymogów redakcyjnych (1p),
2. wykonanie, udokumentowanie oraz opis wykonanych zadań (2p),
3. zastosowanie prawidłowego warsztatu programistycznego (2p).

III. SCENARIUSZ DO ZAJĘĆ

a) ŚRODKI DYDAKTYCZNE

- Sprzętowe:
- komputer,
 - zestaw edukacyjny z mikrokontrolerem ADuC831.
- Programowe:
- zintegrowane środowisko programistyczne języka C dla mikrokontrolerów (Keil uVision).

b) PRZEBIEG ZAJĘĆ

1. Zapoznaj się z informacjami dotyczącymi portów P0, P1, P2, P3.
 - a. Zwróć uwagę na typ portu. Który z portów można skonfigurować jako wejścia lub wyjścia cyfrowe?
 - b. Sprawdzić kiedy dany bit port jest ustawiony jako wejście lub wyjście cyfrowe.
2. Programowa obsługa pojedynczego wejścia i wyjścia mikrokontrolera w symulatorze.
 - a. Zmieniaj cyklicznie wartości binarną pojedynczego wyjścia cyfrowego mikrokontrolera. Wykorzystaj operatory binarne. Sprawdź działanie programu w symulatorze.
 - b. Skonfiguruj pojedynczą linię mikrokontrolera jako wejście cyfrowe. Sprawdź czy ustawiony jest wybrany bit w bajcie. Zmień wartość zmiennej globalnej w zależności od stanu wejścia cyfrowego. Wykorzystaj operatory binarne. Sprawdź działanie programu w symulatorze.
 - c. Przygotuj trzy makrodefinicje: BitUstaw(bajt, nr_bitu), BitKasuj(bajt, nr_bitu) oraz BitSprawdz(bajt, nr_bitu). Odpowiadające odpowiednio za: ustawienie wybranego bitu w bajcie, skanowanie wybranego bitu w bajcie oraz sprawdzenie wybranego bitu w bajcie. Wykorzystaj operatory binarne. Sprawdź działanie programu w symulatorze.
3. Programowa obsługa pojedynczego wejścia i wyjścia mikrokontrolera zestawu edukacyjnego.
 - a. Stan dowolnej diody z Rys. 1 ma odpowiadać stanowi przycisku PR1. Jeśli przycisk jest wciśnięty to dioda jest zapalona, w innym przypadku dioda jest wyłączona. Wykorzystaj napisane makrodefinicje. Sprawdź działanie programu w symulatorze.
 - b. Przygotuj układ do programowania wykonując następujące czynności:
 - i. Włącz zasilanie zestawu edukacyjnego z mikrokontrolerem ADuC831.
 - ii. Podłącz zestaw do komputera z wykorzystaniem złącza RS232.
 - iii. Upewnij się, że masz skonfigurowany projekt według zaleceń z poprzednich zajęć.
 - iv. Upewnij się, że masz skompilowany program.
 - v. Mając wciśnięty przycisk RESET wciśnij przycisk PROGRAM, a następnie kolejno zwolnij przycisk RESET (mignie dioda między przyciskami RESET i PROGRAM) oraz PROGRAM.
 - vi. Wybierz z menu Flash->Download.
 - vii. Przetestuj działanie programu na zestawie edukacyjnym.
 - viii. Każda zmiana programu wymaga powtórzenia etapów od III do VII.
4. Prosty układ kombinacyjny.
 - a. Złożony program wykorzystujący wiele wejść i wiele wyjść cyfrowych.

- i. Wykonaj zadane operacje logiczne na bitach (np. $d=(a \& b|c)$, gdzie $a, b, c, \dots$ są pojedynczymi bitami portu). Sprawdź działanie programu w symulatorze.
 - ii. Wykorzystaj diody z Rys. 1 oraz przycisk PR1 oraz PR2. Napisz makroinstrukcje: ustawiające oraz kasujące stan poszczególnych diod, sprawdzające stan poszczególnych przycisków. Zaproponuj dla wszystkich czterech kombinacji stanu przycisków różne sany diod.
- b. Realizacja sprzętowa.
 - i. Zrealizuj za pomocą bramek logicznych zadanie logiczne przedstawione w punkcie 4.a.
 - ii. Wykorzystaj logiczne przełączniki oraz elementy wyświetlające stan logiczny wyjść.
 - iii. Przetestuj działanie układu w symulatorze.
5. Obsługa przerwań mikrokontrolera.
 - a. Napisz szkielet funkcji obsługującej przerwanie od zdarzenia zewnętrznego – wejście cyfrowe przycisku. Pamiętaj o prawidłowym numerze przerwania.
 - b. Ustaw globalną zgodę na przerwanie oraz zgodę na przerwanie od wybranego zdarzenia zewnętrznego.
 - c. Zmień stan wybranej diody na przeciwny w funkcji obsługującej przerwanie. Sprawdź działanie programu w symulatorze, następnie sprawdź działanie na zestawie edukacyjnym.

IV. PRZYGOTOWANIE DO ZAJĘĆ

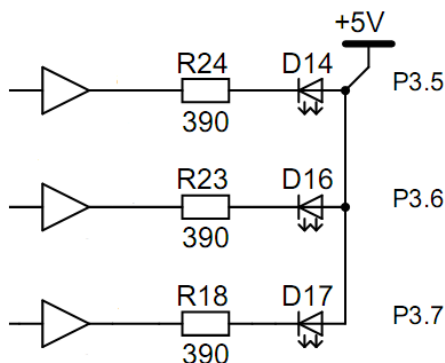
a) ZAPOZNANIE Z PRZEPISAMI BHP

Wszystkie informacje dotyczące instrukcji BHP laboratorium są zamieszczone w sali laboratoryjnej oraz u prowadzącego zajęcia. Wszystkie nieścisłości należy wyjaśnić z prowadzącym laboratorium. Wymagane jest zaznajomienie i zastosowanie do regulaminu.

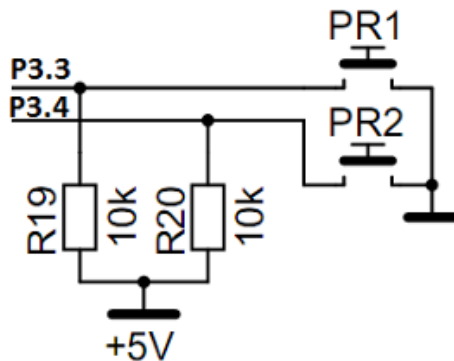
Na zajęcia należy przyjść przygotowanym zgodnie z tematem zajęć. Obowiązuje również materiał ze wszystkich odbytych zajęć.

b) PRZYDATNE SCHEMATY ELEKTRYCZNE

Rys. 1 oraz Rys. 2 przedstawiają fragment schematu elektrycznego zestawu edukacyjnego z mikrokontrolerem ADuC831.



Rys. 1 Schemat połączeń diod



Rys. 2 Podłączenie przycisków monostabilnych do uP

c) WYBRANE REJESTRY MIKROKONTROLERA ADuC831 [1, 2]

Tab. 1 PIN FUNCTION DESCRIPTIONS

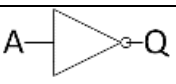
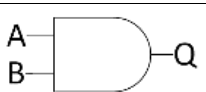
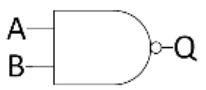
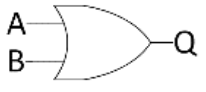
Mnemonic	Type	Function
P1.0–P1.7	I	Port 1 is an 8-bit input port only. Unlike other ports, Port 1 defaults to Analog Input mode, to configure any of these Port Pins as a digital input, write a “0” to the port bit. Port 1 pins are multifunction and share the following functionality.
P3.0–P3.7	I/O	Port 3 is a bidirectional port with internal pull-up resistors. Port 3 pins that have 1s written to them are pulled high by the internal pull-up resistors, and in that state they can be used as inputs. As inputs, Port 3 pins being pulled externally low will source current because of the internal pull-up resistors. Port 3 pins also contain various secondary functions which are described below.
P2.0–P2.7 (A8–A15) (A16–A23)	I/O	Port 2 is a bidirectional port with internal pull-up resistors. Port 2 pins that have 1s written to them are pulled high by the internal pull-up resistors, and in that state they can be used as inputs. As inputs, Port 2 pins being pulled externally low will source current because of the internal pull-up resistors. Port 2 emits the high order address bytes during fetches from external program memory and middle and high order address bytes during accesses to the external 24-bit external data memory space.
P0.7–P0.0 (A0–A7)	I/O	Port 0 is an 8-bit Open Drain Bidirectional I/O port. Port 0 pins that have 1s written to them float, and in that state can be used as high impedance inputs. Port 0 is also the multiplexed low order address and data bus during accesses to external program or data memory. In this application it uses strong internal pull-ups when emitting 1s.

d) PODSTAWOWE OPERACJE LOGICZNE

Funkcje logiczną można opisać za pomocą opisu słownego, tablicy wartości funkcji - tzw. tablicy prawdy (ang. truth table), analitycznie w postaci wyrażenia algebraicznego (zapis matematyczny) lub graficznie za pomocą symboli logicznych. Podstawowe operacje logiczne oraz odpowiadające im bramki logiczne przedstawiono w Tab. 2.

Tab. 2 Podstawowe bramki logiczne

Nazwa	Symbol	Zapis matematyczny	Tabela prawdy
-------	--------	--------------------	---------------

Negacja logiczna (bramka NOT)		$Q = \neg A$ $Q = \bar{A}$	<table><tr><th>A</th><th>Q</th></tr><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td></tr></table>	A	Q	1	0	0	1									
A	Q																	
1	0																	
0	1																	
Koniunkcja (bramka AND)		$Q = A \wedge B$ $Q = A \cdot B$	<table><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Q	0	0	0	0	1	0	1	0	0	1	1	1
A	B	Q																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
Dysjunkcja (bramka NAND)		$Q = \overline{A \wedge B}$ $Q = \overline{A \cdot B}$	<table><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Q	0	0	1	0	1	1	1	0	1	1	1	0
A	B	Q																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
Alternatywa (bramka OR)		$Q = A \vee B$ $Q = A + B$	<table><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	1
A	B	Q																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Binegacja (bramka NOR)		$Q = \overline{A \vee B}$ $Q = \overline{A + B}$	<table><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Q	0	0	1	0	1	0	1	0	0	1	1	0
A	B	Q																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Alternatywa wykluczająca (bramka XOR)		$Q = A \underline{\vee} B$ $Q = A \oplus B$ $Q = (A \cdot \bar{B}) + (\bar{A} \cdot B)$	<table><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	0
A	B	Q																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Negacja alternatywy wykluczającej (bramka XNOR)		$Q = \overline{A \underline{\vee} B}$ $Q = \overline{A \oplus B}$	<table><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Q	0	0	1	0	1	0	1	0	0	1	1	1
A	B	Q																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

e) ALGEBRA BOOLE'A

Podstawowe prawa i tożsamości algebry Boole'a zostały przedstawione w Tab. 3. Dodatkowo należy zapamiętać, że podwójne przeczenie nie zmienia wartości wyrażenia logicznego np. $\bar{\bar{A}} = A$ lub $\bar{\bar{A} \cdot \bar{B}} = A \cdot B$. Dodatkowo wykorzystując prawo De Morgana można uzyskać następujące zależności $A \cdot B = \overline{\bar{A} \cdot \bar{B}} = \bar{\bar{A}} + \bar{\bar{B}}$ oraz $A + B = \overline{\bar{A} + \bar{B}} = \bar{\bar{A}} \cdot \bar{\bar{B}}$.

Tab. 3 Podstawowe prawa algebry Boole'a

Rodzaj	Nazwa	Wzór odniesiony do iloczynu logicznego	Wzór odniesiony do sumy logicznej
Prawa	przemienności	$A \cdot B = B \cdot A$	$A + B = B + A$
	łączności	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$

	rozdzielczości	$A \cdot (B + C) = A \cdot B + A \cdot C$	$A + B \cdot C = (A + B) \cdot (A + C)$
	De Morgana	$\overline{A \cdot B} = \bar{A} + \bar{B}$ $\overline{A \cdot B \cdot C \cdot \dots} = \bar{A} + \bar{B} + \bar{C} + \dots$	$\overline{A + B} = \bar{A} \cdot \bar{B}$ $\overline{A + B + C + \dots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \dots$
Tożsamości	podstawowe	$A \cdot 0 = 0$ $A \cdot 1 = A$ $A \cdot A = A$ $A \cdot \bar{A} = 0$	$A + 0 = A$ $A + 1 = 1$ $A + A = A$ $A + \bar{A} = 1$

f) OPERATORY BITOWE W JĘZYKU C

Język C wyposażony jest w operatory bitowe. Symbole oraz ich nazwy przedstawia Tab. 4.

Tab. 4 Operacje na bitach

Symbol	Nazwa
~	Bitowa negacja
	Bitowa alternatywa (OR)
&	Bitowa koniunkcja (AND)
^	Bitowa różnica symetryczna (XOR)
>>	Bitowe przesunięcie w prawo
<<	Bitowe przesunięcie w lewo

Przypomnienie: Bit może przyjąć jedną z dwóch wartości: 0 lub 1. Gdy ma on wartość 1 mówimy, że jest ustawiony. Należy również pamiętać, że operacje bitowe dotyczą działań na pojedynczych bitach!

▪ Bitowa negacja (~)

Operator bitowej negacji ~ jest jednoargumentowa i zmienia wartość bitów według zasad umieszczonych w Tab. 5.

Tab. 5 Bitowa negacja

a	~a
0	1
1	0

Za przykład weźmy liczbę 25_{10} zapisaną w systemie binarnym:

$$25_{10} = 0x19_{16} = 00011001_2$$

Zapisana została ona na ośmiu bitach. Po wykorzystaniu operatora ~ otrzymano:

$$\sim(25)_{10} = \sim(00011001)_2 = 11100110_2$$

Zera z liczby 25_{10} zostały zastąpione jedynkami i na odwrót.

▪ Bitowa alternatywa (|)

Jest to operacja dwuargumentowa, która działa zgodnie z zasadami zawartymi w Tab. 6

Tab. 6 Bitowa alternatywa

a	b	a b
0	0	0
0	1	1
1	0	1
1	1	1

Elementem neutralnym alternatywy jest 0. Przykład liczba 25_{10} oraz 6_{10} , zapisane w systemie binarnym:

$$25_{10} = 0x19_{16} = 00011001_2$$

$$6_{10} = 0x06_{16} = 00000110_2$$

Po zastosowaniu operatora | otrzymujemy:

$$(00011001)_2 | (00000110)_2 = 00011111_2$$

stosując inny zapis:

$$\begin{array}{r} 00011001_2 \\ | \\ 00000110_2 \\ \hline 00011111_2 \end{array}$$

▪ Bitowa koniunkcja (&)

Jest to operacja dwuargumentowa, która działa zgodnie z zasadami zawartymi w Tab. 7.

Tab. 7 Bitowa koniunkcja

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

Przykład 19_{10} oraz 85_{10} , zapisane w systemie binarnym:

$$19_{10} = 0x13_{16} = 00010011_2$$

$$85_{10} = 0x55_{16} = 01010101_2$$

Po wykorzystaniu operatora & otrzymujemy:

$$(00010011)_2 \& (01010101)_2 = 00010001_2$$

stosując inny zapis:

$$\begin{array}{r} 00010011_2 \\ \& 01010101_2 \\ \hline 00010001_2 \end{array}$$

▪ Bitowa różnica symetryczna/alternatywa wykluczająca (^)

Jest to operacja dwuargumentowa, która działa zgodnie z zasadami zawartymi w Tab. 8.

Tab. 8 Bitowa różnica symetryczna

a	b	a [^] b
0	0	0
0	1	1
1	0	1
1	1	0

Przykład 45₁₀ oraz 25₁₀, zapisane w systemie binarnym:

$$45_{10} = 0x2D_{16} = 00101101_2$$

$$25_{10} = 0x19_{16} = 00011001_2$$

Po wykorzystaniu operatora ^ otrzymujemy:

$$(00101101)_2 \wedge (00011001)_2 = 00110100_2$$

stosując inny zapis:

$$\begin{array}{r} 00101101_2 \\ \wedge \quad 00011001_2 \\ \hline 00110100_2 \end{array}$$

▪ Bitowe przesunięcie w prawo (>>)

Jest to operacja dwuargumentowa, którą jest zapisywana w następujący sposób:

$$a \gg b$$

Powyższy zapis jest interpretowany następująco: *przesuń liczbę a o b bitów w prawo*. Operacja odpowiada podzieleniu liczby a przez 2^b, gdzie wykładnik potęgi b oznacza liczbę przesunięć. Należy zapamiętać, że na najstarszą pozycję dopisywany jest bit/bity o wartości zero, natomiast najmłodszy bit/bity są tracone. Przykładowo liczba 48₁₀ przesunięta o 2 bity w prawo:

$$a = 48_{10} = 0x30_{16} = 00110000_2$$

$$(a \gg 2) = (00110000_2 \gg 2) = 0001100_2 = 12_{10}$$

Zgodnie z definicją przesunięcia bitowego w prawo, po przesunięciu liczby 48₁₀ o 2 bity w prawo otrzymano liczbę $\frac{48}{2^2} = \frac{48}{4} = 12$. Na niebiesko zaznaczone zostały bity dopisywane na najstarszych pozycjach.

▪ Bitowe przesunięcie w lewo (<<)

Jest to operacja dwuargumentowa, którą jest zapisywana w następujący sposób:

$$a \ll b$$

Powyższy zapis jest interpretowany następująco: *przesuń liczbę a o b bitów w lewo*. Operacja odpowiada pomnożeniu liczby a przez 2^b, gdzie wykładnik potęgi b oznacza liczbę przesunięć. Należy zapamiętać, że na najmłodsza pozycję dopisywany jest bit/bity o wartości zero, natomiast najstarszy bit/bity są tracone. Przykład liczba 48₁₀ jest przesunięta o 2 bity w lewo:

$$a = 48_{10} = 0x30_{16} = 00110000_2$$

$$a \ll 2 = (00110000_2 \ll 2) = 11000000_2 = 192_{10}$$

Zgodnie z definicją przesunięcia bitowego w lewo, po przesunięciu liczby 4810 o 2 bity w lewo otrzymano liczbę $48 \cdot 2^2 = 48 \cdot 4 = 192$. Na niebiesko zaznaczone zostały bity dopisywane na najmłodsze pozycje.

g) OPERATORY LOGICZNE W JĘZYKU C

Język C wyposażony jest w operatory logiczne, których symbole oraz nazwy przedstawia Tab. 9.

Tab. 9 Operacje logiczne

Symbol	Nazwa	Przykład	Zapis matematyczny
!	Logiczna negacja	!a	$\neg a$ lub $\bar{a}$
	Logiczna alternatywa (OR)	a b	$a b$ lub $a \vee b$
&&	Logiczna koniunkcja (AND)	a && b	$a \& b$ lub $a \wedge b$

Operatory logiczne mają na celu tworzenie złożonych wyrażeń logicznych.

h) WSTĘP DO PRZERWAŃ

Przerwanie (ang. *interrupt*) – jest to sygnał powodujący wstrzymanie dotychczasowego programu i wykonanie odpowiedniej procedury obsługujące przerwanie. W programie należy udzielić zgody na przerwanie. Wyszczególniamy dwa poziomy pozwoleń. Zgoda globalna zezwalająca lub blokująca ogólną obsługę przerw. Zgoda lokalna zezwalająca na obsługę przerwania od wybranego zdarzenia. Przykładowo

EX1=1; zgoda na przerwania od INT1
EA=1; zgoda globalna na przerwania

Wzorzec definicji funkcji obsługującej przerwanie umieszczamy przed main():

```
void NAZWA_FUNKCJI (void) interrupt NUMER_PRZERWANIA
{
// Miejsce na kod
}
```

Obsługa przerwania od danego zdarzenia wymaga znajomości jego numeru przerwania Tab. 10 [1, 2].

Tab. 10 Numery przerw

Źródło	Adresy	Numer przerwania
IE0	0003H	0
TF0	000BH	1
IE1	0013H	2
TF1	001BH	3
RI + TI	0023H	4
TF2+EXF2	002BH	5
ADCI	0033H	6
I2CI+ISPI	003BH	7
PSMI	0043H	8
TII	0053H	10
WDS	005BH	11

Istotnym zagadnieniem jest priorytet wykonywania przerw, gdy te występują jednocześnie. Dokładna tabela priorytetów znajduje się w dokumentacji ADUC831 [1].

i) SZKIELET PROGRAMU

W trakcie zajęć należy korzystać z poniższego szablonu programu:

```
#include <aduc831.h> //zawiera definicje rejestrów mikrokontrolera ADuC831
//Miejsce na makroinstrukcje i definicje
//Miejsce na zmienne globalne

int main(void)
{
//Miejsce na zmienne lokalne
//Miejsce na inicjalizację
while(1) //Pętla nieskończona
{
//Miejsce na kod programu głównego
}
return 0;
}
```

LITERATURA

1. ADUC831 datasheet and product info | Precision Analog Microcontroller: 1.3MIPS 8052 MCU + 62kB Flash + 8-Ch 12-Bit ADC + Dual 12-Bit DAC | Analog Microcontrollers | Analog Devices [online]. [udostępniono 19.11.2014]. Pobrano: <http://www.analog.com/en/processors-dsp/analog-microcontrollers/aduc831/products/product.html>.
2. ADuC831 MicroConverter, 12-Bit ADCs and DACs with Embedded 62 kBytes Flash MCU Data Sheet, (Rev. 0) - ADUC831.pdf [online]. s.l.: s.n. [udostępniono 19.11.2014]. Pobrano: http://www.analog.com/static/imported-files/data_sheets/ADUC831.pdf.