Temat 7. Wprowadzenie do programowania mikrokontrolerów na przykładzie Microchip PIC16

Spis treści do tematu 7

7.1. Wprowadzenie

Mikrokontroler (skrót ang. MCU lub μC) – autonomiczny system mikroprocesorowy zrealizowany w formie pojedynczego układu scalonego, zawierającego wszystkie podstawowe bloki, w tym co najmniej:

- jednostkę centralną (CPU),
- pamięć programu,
- pamięć danych RAM,
- zegar systemowy i układy czasowe,
- układy wejścia-wyjścia.

Główne obszary zastosowań mikrokontrolerów:

- sterowanie sprzętem AGD i RTV,
- sterowanie podzespołami komputerów klasy PC,
- przemysłowe układy automatyki,
- systemy telekomunikacyjne,
- instalacje alarmowe,
- podzespoły samochodowe,
- zabawki i gadżety elektroniczne,
- układy kontrolno-pomiarowe.

Historia

1971 – Intel 4004
pierwszy kompletny mikroprocesor CPU (ang. *central processing unit*) w jednym układzie scalonym. 4-bitowa jednostka arytmetyczno-logiczna. Opracowano także współpracujące układy pamięci ROM, RAM i układy wejścia/wyjścia.

1971 - Texas Instruments TMS0100
pierwszy 4-bitowy mikrokontroler w jednym układzie scalonym, przeznaczony do kalkulatorów kieszonkowych.

1975 – General Electric PIC1650
Mikrokontroler 8-bitowy z którego wywodzi się liczna rodzina układów PIC produkowanych do dziś przez Microchip Technology Inc.

1976 – Intel 8048
pierwszy 8-bitowy masowo produkowany mikrokontroler.

1980 – Intel 8051
mikrokontroler 8-bitowy, który stał się powszechnym standardem. Wielu producentów produkuje do dziś różne odmiany tego MCU (np. Atmel Corp.).

1993 – Microchip 16C84
pierwszy mikrokontroler z kasowaną elektrycznie pamięcią EEPROM. Wcześniej stosowano pamięci EPROM kasowane ultrafioletem oraz ROM.

Aktualne trendy na rynku MCU

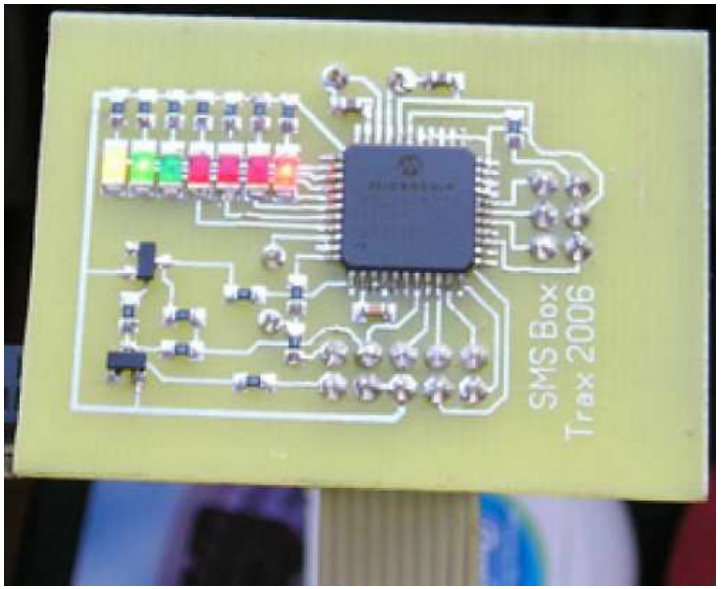
1). Od kilkunastu lat rozszerza się oferta mikrokontrolerów 16- i 32-bitowych ogólnego przeznaczenia oraz DSP (ang. *digital signal processing*), jednakże 8-bitowe MCU są wciąż rozwijane i stanowią większość sprzedawanych MCU.

2). Na rynku MCU obecnych jest wielu producentów, którzy oferują liczne rozwiązania niezgodne pod względem oprogramowania. Mikrokontrolery zgodne z Intel 8051 są wciąż obecne na rynku, jednakże są już uważane za przestarzałe.

3). Nacisk na ograniczanie zużycia energii, np. technologia XLP (eXtreme Low Power) stosowana w nowych konstrukcjach firmy Microchip ogranicza pobór prądu do około 30 μA/MHz albo 9 nA w trybie uśpienia.

4). Tendencja do coraz większej integracji w MCU licznych dodatkowych układów:
- sprzętowe kontrolery wyświetlaczy LCD, magistral RS232, RS485, I^2C, SPI, USB,
- kilka układów zegarów do pomiarów liczby/czasu impulsów,
- PWM (układy impulsowego sterowania mocą),
- przetworniki A/D i D/A,
- pomocnicze układy analogowe (komparatory, selektory, źródła napięć odniesienia).

Mikroprocesor czy mikrokontroler?



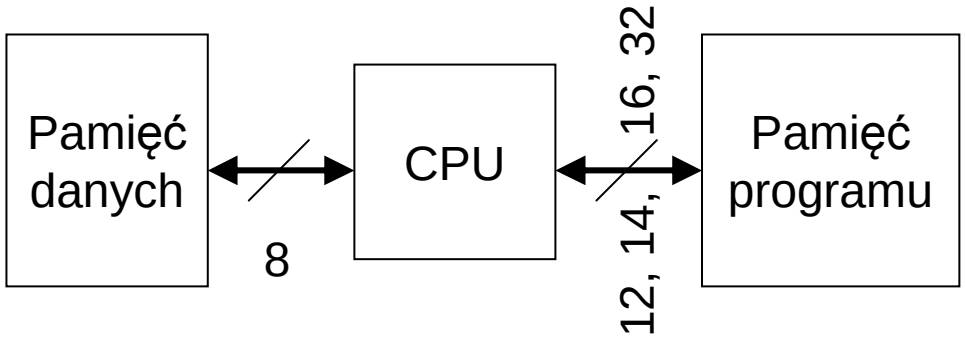
Mikroprocesory (CPU) zapewniają:
- najwyższą moc obliczeniowa,
- bezpośrednie adresowanie dużych pamięci operacyjnych,

Mikroprocesory wymagają jednak użycia wielu dodatkowych, kosztownych elementów.

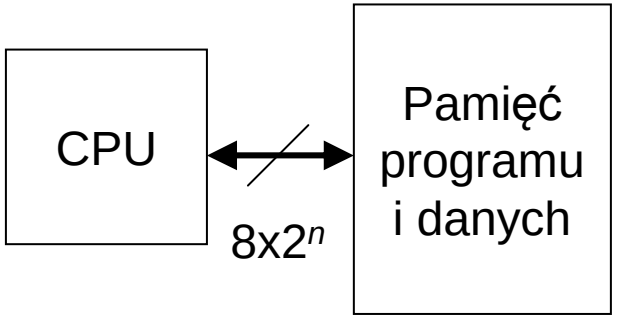
Mikrokontrolery (MCU) zapewniają:
- niską cenę,
- krótki czas opracowania urządzenia,
- prostotę i małe rozmiary urządzenia,
- niezawodność i trwałość,
- mały pobór prądu,
- wielki wybór modeli MCU.

Architektura urządzeń mikroprocesorowych

Rys. 7.1. Architektura Harvard



Rys. 7.2. Architektura von-Neumanna



Architektura typowa dla małych MCU

- Oddzielne pamięci oraz przestrzenie adresowe dla danych i dla kodu programu.
- Jednoczesne przesłania po magistralach obu pamięci.
- 1 instrukcja zwykle przypada na jedno słowo w pamięci programu.
- Pamięć danych zwykle jest mała względem pamięci programu ale o dużej szybkości (typu SRAM). Pamięć podręczna zwykle nie występuje.

Architektura komputerów klasy PC

- Wspólna pamięć danych i kodu programu umożliwia optymalny przydział zasobów.
- Instrukcje wielobajtowe, często o różnych rozmiarach.

Ograniczenia tej architektury są łagodzone przez stosowanie pamięci podręcznych (Cache RAM) osobnych dla kodu i dla danych

7.2. Architektura mikrokontrolerów PIC10…18

Mikrokontrolery PIC są zbudowane wg zmodyfikowanej architektury Harvard RISC

- Osobne pamięci programu i danych (Architektura Harvard),
- Zredukowany zestaw instrukcji (RISC),
- Jedna instrukcja zajmuje jedno słowo maszynowe o długości 12, 14 albo 16 bitów,
- Przetwarzanie potokowe dwóch instrukcji: (1) pobieranie instrukcji + (2) wykonywanie,
- Ustalony czas wykonywania instrukcji (tylko skoki dwa razy dłużej).

Cechy nietypowe architektury mikrokontrolerów PIC

- Architektura pliku rejestrów (*ang. register file architecture*) – rejestry o specjalnym znaczeniu (np. rejestry układów wej./wyj., licznik instrukcji) są włączone w przestrzeń adresową i można je adresować jak komórki danych w pamięci RAM.
- Ortogonalność - większość operacji można wykonać na dowolnej komórce danych w pamięci RAM a także na rejestrach specjalnych, włączając w to licznik rozkazów i rejestr STATUS (m.in. znaczniki wyników operacji arytm.-logicznych).
- Istnieje tylko jeden rejestr danych W pełniący rolę drugiego (oprócz komórki pamięci) argumentu instrukcji.
- Stos sprzętowy leży poza dostępną przestrzenią adresową danych i może przechowywać tylko adresy powrotne z procedur.

Tabela 7.1. Porównanie rodzin 8-bitowych mikrokontrolerów PIC®

Architektura	Baseline Arch.	Midrange Arch.	Enhanced Midrange Arch.	PIC18 Arch. (High Performance)
liczba złącz	6 – 40	8 – 64	8 – 64	18 – 100
wydajność	do 5 MIPS	do 5 MIPS	do 12 MIPS	do 32 MIPS
przerwania	brak	jednopoziomowe	jednopoziomowe	wielopoziomowe
instrukcje	33, 12-bitowe	35, 14-bitowe	49, 14-bitowe	75-83, 16-bitowe
Pamięć kodu	do 2 K słów, banki 512 słów	do 8 K słów, banki 2 K słów	do 32 K słów, banki 2 K słów	do 64K słów, jeden bank*
Pamięć danych	do 134 B, banki 32 B	do 368 B, banki 128 B	do 4 KB, banki 128 B	do 13 KB, banki 256 B
stos adresów	2 poziomy	8 poziomów	16 poziomów	32 poziomy
opis	najniższy koszt, najmniejsze wymiary	optymalny koszt do możliwości, zintegrowane układy ADC, SPI, I^2C, UART, PWM i in.	optymalizowany dla języka C	optymalizowany dla języka C, liczne układy peryferyjne m.in. USB, Ethernet
symbole katalogowe	PIC10, PIC12, (wybrane PIC16)	PIC16, (wybrane PIC12)	PIC12F1xxx, PIC16F1xxx	PIC18

* - dla adresu bezpośredniego w kodzie instrukcji GOTO, CALL.

Tabela 7.2. Związek oznaczeń katalogowych 8-bitowych mikrokontrolerów PIC® z architekturą wewnętrzną oraz z liczbą pinów w obudowie.

Architektura	Symbol katalogowy	Liczba aktywnych pinów w obudowie									
		6	8	14	18	20	28	40	64	80	100
High Perf.	PIC18___										
Enhanced Midrange	PIC16F1xxx										
	PIC12F1xxx										
Midrange	PIC16___										
	PIC12F6xx										
Baseline	PIC16F5x(x)										
	PIC12___										
	PIC10___										

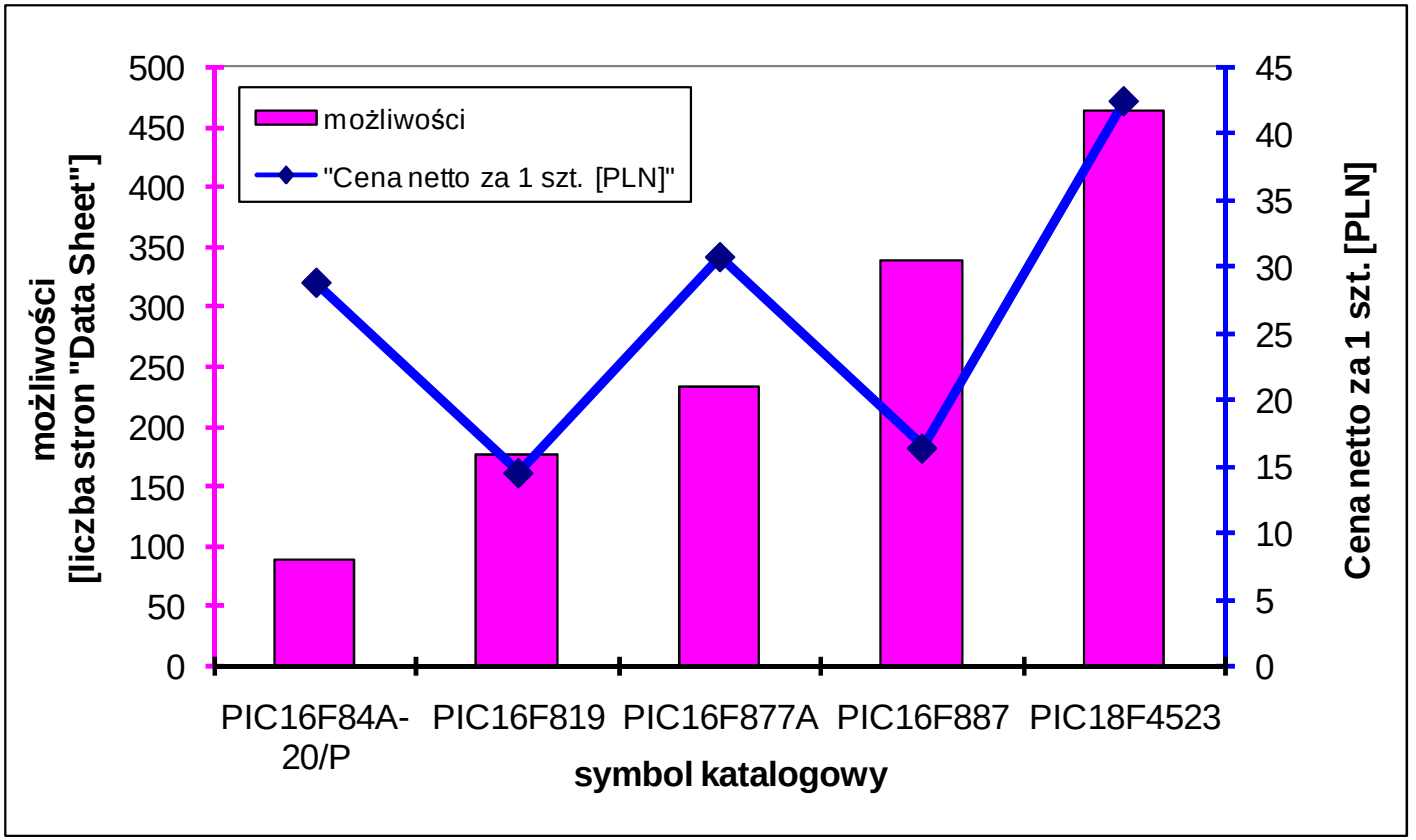
Porównanie wybranych mikrokontrolerów z rodziny „Midrange”

PIC16F84A – jeden z najstarszych MCU w rodzinie Midrange; wyznacza dolny kraniec możliwości rodziny (w 1993r. model 16C84 z pamięcią kodu EEPROM),
PIC16F819 – średnie możliwości,
PIC16F877A – układ o względnie bogatym wyposażeniu w rodzinie Midrange.

Tabela 7.3. Zestawienie wyposażenia wybranych mikrokontrolerów Midrange.

Wyposażenie	Mikrokontroler		
	PIC16F84A	PIC16F819	PIC16F877A
Pamięć programu (słowa 14-bitów)	1024	2048	8192
Pamięć danych SRAM (bajty)	68	256	368
Pamięć danych EEPROM (bajty)	64	256	256
Wejścia/wyjścia cyfrowe	13	do 16	do 33
Oscylator taktujący zewn./wewn.	tak / –	tak / tak	tak / –
Przetwornik 10 bitowy A/D	–	1	1
Zegary 8/16 bitowe	1 / 0	2 / 1	2 / 1
Komparatory analogowe	–	2	2
Układy CCP i PWM	–	1	1
Interfejsy transmisji szeregowej	–	SPI, I^2C (Slave)	USART, SPI, I^2C (Master/Slave)
8-bitowy port równoległy	–	–	PSP

Cena mikrokontrolera nie jest zdeterminowana przez możliwości !

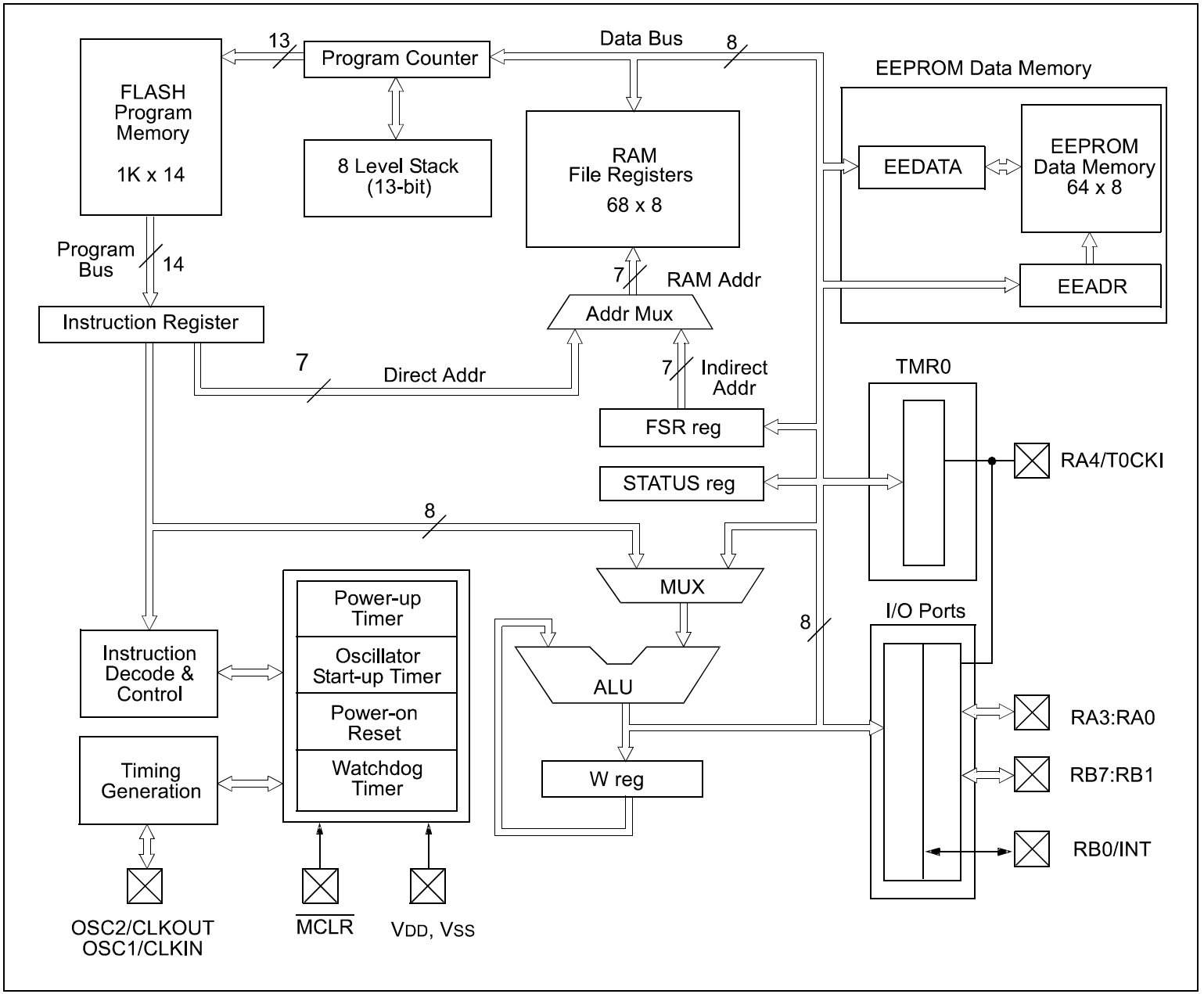


Ceny netto za 1szt. na postawie:
www.tme.pl 2 kwietnia 2025 r.

Rys. 7.3. Zestawienie cen i możliwości wybranych mikrokontrolerów PIC.

Cena PIC16F84A wynika głównie z jego sukcesu rynkowego i długiej obecności na rynku. Dostępne są zamienniki o większych możliwościach, zgodne pod względem układu wyprowadzeń i parametrów elektrycznych ale program wymaga dostosowania.

PIC16F84A można zastąpić układem PIC16F819,
PIC16F877A można zastąpić jednym z układów PIC16F887 lub PIC18F452x.



Rys. 7.4. Schemat blokowy mikrokontrolera PIC16F84A pochodzący z dokumentacji producenta (ang. *Data Sheet*) [2]. Opis skrótów na następnej stronie.

Alfabetyczny opis ważniejszych skrótów z rys. 7.4. „Schemat blokowy mikrokontrolera PIC16F84A”

ALU – jednostka arytmetyczno-logiczna (*ang. Arithmetic Logic Unit*).

EEPROM – pamięć nieulotna kasowana przy użyciu prądu elektrycznego (*ang. electrically erasable programmable read-only memory*). W układach PIC16Fxx pamięć ta może być kasowana i zapisywana przez program operujący na rejestrach EEDATA i EEADR w celu zachowania danych podczas braku zasilania. Mała szybkość w porównaniu do RAM i FLASH.

FLASH – pamięć nieulotna będąca rozwinięciem EEPROM o większej szybkości. W układach PIC16Fxx pamięć ta przechowuje kodu programu i jest zapisywana przez zewnętrzny programator.

I/O Ports – porty wejścia/wyjścia (*ang. Input/Output ports*) połączone bezpośrednio z wyprowadzeniami układu.

MCLR – (*ang. master clear*), podawany z zewnętrz sygnał resetu mikrokontrolera.

MUX – multiplekser (*ang. multiplexer*) wybierający źróło danych dla ALU,

RAM – szybka pamięć o dostępie swobodnym (*ang. random-access memory*) używana do przechowywania danych programu.

TMR0 – zegar numer 0 (*ang. timer 0*), który można wykorzystać w programie do pomiaru czasu i okresowego generowania przerwań.

W reg. – rejestr W używany w operacjach arytmetycznych i logicznych jako źródło jednej z danych, opcjonalnie może być także celem wyniku operacji.

Narzędzia programistów dla mikrokontrolerów PIC®

Producent mikrokontrolerów PIC dostarcza dwa zintegrowane środowiska projektowe MPLAB IDE (Windows) oraz MPLAB IDE X (Windows + Linux, brak obsługi starszych programatorów w tym brak PICSTART Plus, brak kompilatora C dla PIC10/12/16).
Pakiet MPLAB IDE zawiera m.in:

- edytor tekstu do edycji plików źródłowych i nagłówkowych,
- menadżer projektów ułatwiający organizację pracy z projektami zawierającymi wiele plików,
- MPASM – makroasembler dla wszystkich mikrokontrolerów firmy Microchip,
- MPLINK – program łączący moduły tworzone za pomocą asemblera i kompilatorów języka C w jeden plik z danymi dla programatora, emulatora lub symulatora,
- MPLAB SIM – programowy symulator mikrokontrolerów,
- obsługę zewnętrznych urządzeń programatorów i emulatorów sprzętowych, m.in. programator PICSTART Plus dostępny w Laboratorium Techniki Cyfrowej,
- system pomocy i pliki z dokumentacją.

Dla mikrokontrolerów PIC z rodzin Baseline i Midrange producent nie dostarcza własnego kompilatora języka C, jednakże instalator pakietu MPLAB IDE umożliwia zainstalowanie kompilatorów innych firm, np.:

- HI-Tech C for the PIC 10/12/16 MCU Family,
- CCS C Compiler for PIC 12/14/16/18.

Pakiet MPLAB IDE jest udostępniany bezpłatnie przez firmę Microchip na stronie internetowej **www.microchip.com**

Przenoszenie programu między różnymi modelami 8-bitowych mikrokontrolerów PIC® w ramach jednej rodziny

Mikrokontrolery należące do jednej rodziny (Baseline, Midrange, Enhanced Midrange albo PIC18) są zgodne pod względem listy instrukcji, trybów adresowania pamięci, rozmiarów banków pamięci i budowy stosu adresów powrotnych.

Zamiana danego mikrokontrolera na inny z tej samej rodziny i zgodny pod względem układu wyprowadzeń i parametrów elektrycznych zazwyczaj wymaga jednak dostosowania programu. Należy uwzględnić:

1). Różnice w mapach pamięci dla różnych układów MCU.
2). Różnice w rejestrze konfiguracyjnym (zapisywany z kodem programu, adres 2007h).
3). Różnice w trybach działania oscylatora taktującego pracę MCU.
4). Różnice w budowie portów wej./wyj. (np. linia RA4 skonfigurowana jako wyjście w PIC16F84A jest typu „otwarty dren”; w PIC16F819 typu „push-pull” z możliwością programowej redukcji do „otwartego drenu”).
5). Niektóre wyprowadzenia, oprócz funkcji wej./wyj. w standardzie TTL, mogą mieć także inne funkcje zależne od typu układu (np. wejścia komparatorów analogowych, przetworniki A/D i D/A). Funkcje ustawione domyślnie po załączeniu zasilania mogą być nieodpowiednie i trzeba je zmienić programowo.

Przenoszenie programu między 8-bitowymi mikrokontrolerami PIC® z różnych rodzin

Język maszynowy 8-biowych mikrokontrolerów PIC należących do różnych rodzin (Baseline, Midrange, Enhanced Midrange i PIC18) wykazuje daleko idące podobieństwa.

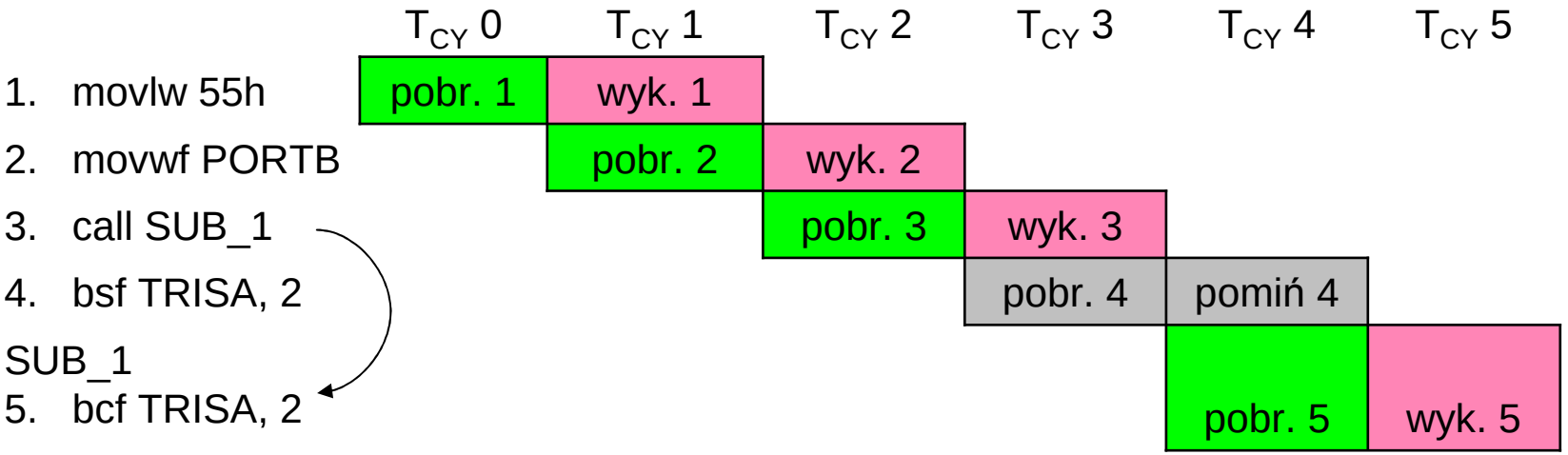
Migracja „do góry”
Instrukcje zaimplementowane w niższej rodzinie 8-bitowych mikrokontrolerów PIC są dostępne także w wyższych rodzinach, co upraszcza migrację „do góry”. Jest to zgodność tylko na poziomie tekstu źródłowego z instrukcjami, a nie pełna zgodność kodowania instrukcji. Ponadto należy także rozważyć różnice wyliczone wcześniej podczas omawiania migracji między MCU należącymi do jednej rodziny.

Migracja „w dół” jest trudniejsza gdyż:
- lista używanych instrukcji musi zostać ograniczona,
- głębokość sprzętowego stosu dla adresów powrotnych z procedur ulega zmniejszeniu,
- rozmiary banków pamięci dla danych i kodu ulegają zmniejszeniu,
- względne adresowanie pamięci, które zaimplementowano efektywnie w rodzinach Enhanced Midrange i PIC18, wymaga większej liczby instrukcji w rodzinach Baseline i Midrange.

7.3. Rozkazy mikrokontrolerów PIC Midrange

Wszystkie mikrokontrolery z rodziny Midrange mają taką samą listę 35 instrukcji (nie dotyczy *Enhanced Midrange*). Jeden rozkaz zajmuje zawsze jedno słowo 14-bitowe.

Wykonywanie instrukcji przebiega w cyklu (T_{CY}) złożonym z czterech taktów zegara:
Q1: dekodowanie instrukcji lub warunkowe pominięcie operacji,
Q2: odczyt danych lub brak operacji,
Q3: przetwarzanie danych,
Q4: zapis wyniku lub brak operacji.



Rys. 7.5. Potokowe przetwarzanie instrukcji: Podczas wykonywania jednej instrukcji pobierana jest już następna instrukcja. W przypadku skoków (np. call) wyjątkowo potrzebne są 2 cykle instrukcji (8 zegarowych).

Operacje arytmetyczno-logiczne mają dwa argumenty:

1). Rejestr W, który nie jest dostępny w przestrzeni adresowej,

2). Komórka pamięci RAM albo stała w kodzie instrukcji.

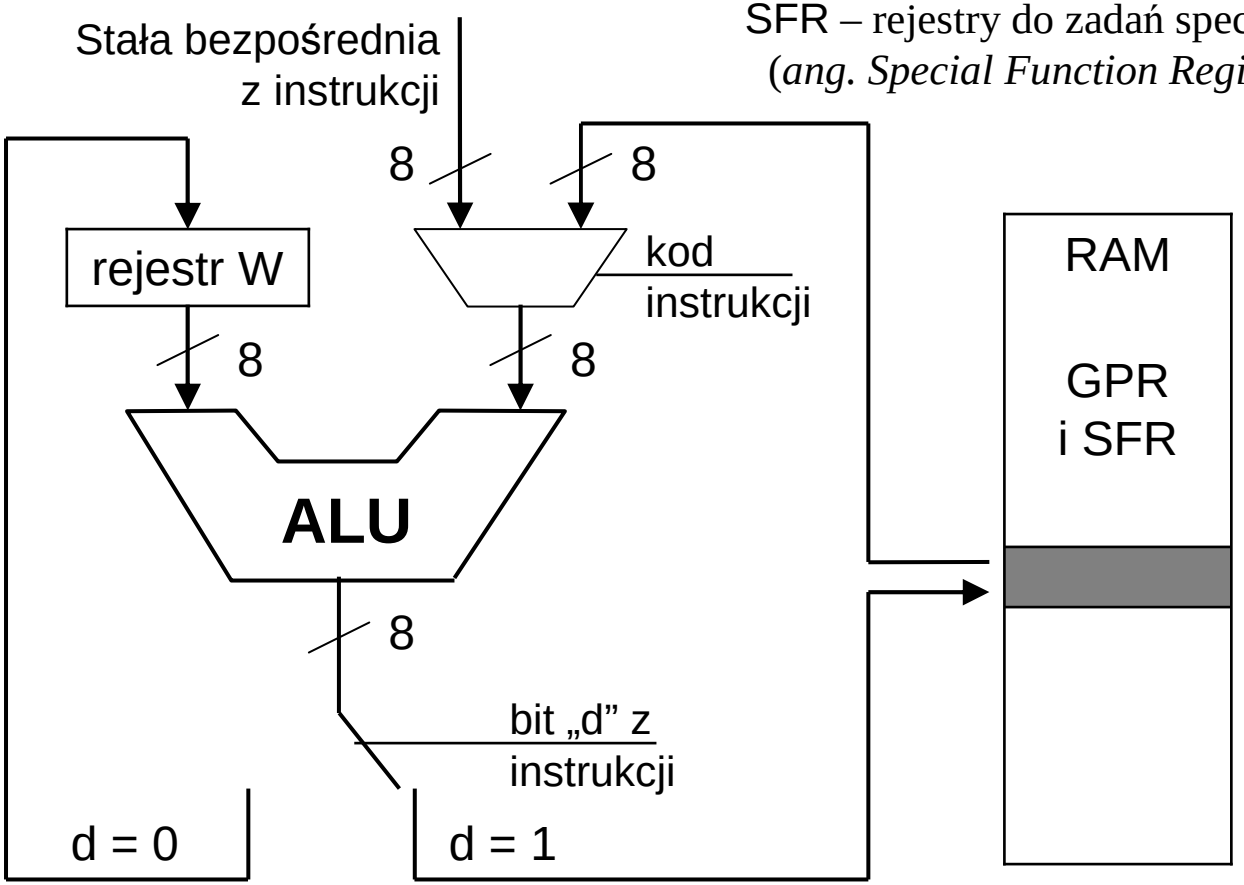
Wynik operacji może zostać zapisany do:

1). Rejestru W,

2). Komórki pamięci RAM.

GPR – komórki pamięci RAM ogólnego przeznaczenia (*ang. General Purpose Registers*),

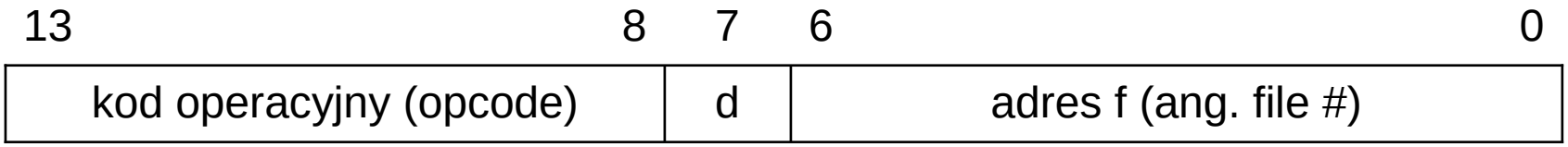
SFR – rejestry do zadań specjalnych (*ang. Special Function Registers*).



albo stała w instrukcji

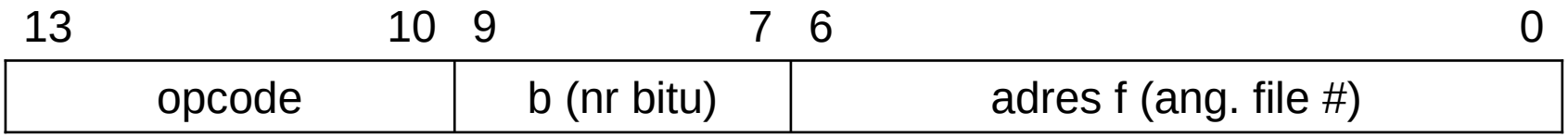
Rys. 7.6. Schemat blokowy otoczenia jednostki arytmetyczno-logicznej (ALU).

Rozkazy operujące na bajtach w pamięci

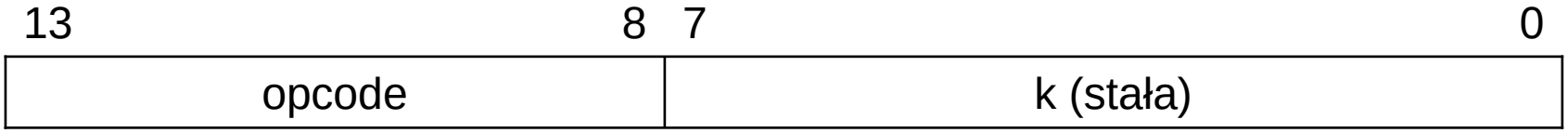


d – wynik zapisz do rej. W albo RAM

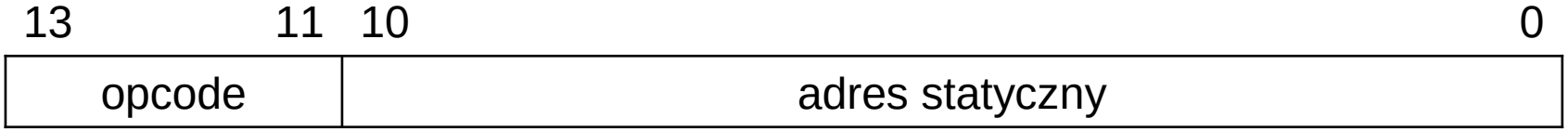
Rozkazy operujące na bitach



Rozkazy operujące na stałych



Skoki call i goto



Rozkazy bez argumentów - brak wyróżnionych pól (instrukcje CLRW, CLRWDT, NOP, RETFIE, RETURN, SLEEP, OPTION).

Pominięto specjalny format instrukcji TRIS uważanej obecnie za przestarzałą.

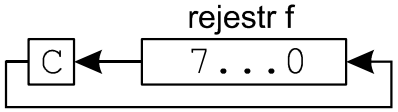
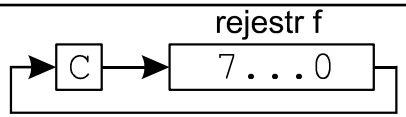
Tabela 7.4. Rozkazy operujące na bajtach w pamięci

Mnemonik, argument	Opis	Funkcja	Zmieniane znaczniki
ADDWF f,d	Dodaj W i f	W + f → d	C, DC, Z
ANDWF f,d	Bitowy iloczyn logiczny W i f	W AND f → d	Z
CLRF f	Wyzeruj f	0 → f	Z
CLRW	Wyzeruj W	0 → W	Z
COMF f,d	Zaneguj bity f	NOT f → d	Z
DECF f,d	Zmniejsz f o 1	f – 1 → d	Z
DECFSZ f,d	Zmniejsz f o 1, pomiń następny rozkaz jeśli wynik=0	f – 1 → d, jeśli d=0 to nic nie rób podczas następnego rozkazu	–
INCF f,d	Zwiększ f o 1	f + 1 → d	Z
INCFSZ f,d	Zwiększ f o 1, pomiń następny rozkaz jeśli wynik=0	f + 1 → d, jeśli d=0 to nic nie rób podczas następnego rozkazu	–

d – bit wyboru przeznaczenia wyniku:
d=0: zapisz wynik w rejestrze W,
d=1: zapisz wynik w pamięci pod adresem f,
f – zmienna dana 7-bitowym adresem w pamięci.

Znaczniki w rej. STATUS:
C – przeniesienie,
DC – przeniesienie połówkowe,
Z – zero.

Rozkazy operujące na bajtach w pamięci - kontynuacja

Mnemonik, argument	Opis	Funkcja	Zmieniane znaczniki
IORWF f,d	Bitowa suma logiczna W i f	W OR f → d	Z
MOVF f,d	Prześlij f do d	f → d	Z
MOVWF f	Prześlij W do f	W → f	–
NOP	Nic nie rób	–	–
RLF f,d	Przesuń f cyklicznie w lewo przez znacznik C		C
RRF f,d	Przesuń f cyklicznie w prawo przez znacznik C		C
SUBWF f,d	Odejmij W od f	f – W → d	C,DC,Z
SWAPF f,d	Zamień tetrady w f	f<0:3> ↔ f<4:7>	–
XORWF f,d	Bitowa suma modulo 2 na W i f	W XOR f → d	Z

d – bit wyboru przeznaczenia wyniku:
d=0: zapisz wynik w rejestrze W,
d=1: zapisz wynik w pamięci pod adresem f,
f – zmienna dana 7-bitowym adresem w pamięci.

Znaczniki w rej. STATUS:
C – przeniesienie,
DC – przeniesienie połówkowe,
Z – zero.

Tabela 7.5. Rozkazy operujące na bitach

Mnemonik, argument	Opis	Funkcja	Zmieniane znaczniki
BCF f,b	Wyzeruj bit b w bajcie f	0 → f<b>	–
BSF f,b	Ustaw bit b w bajcie f	1 → f<b>	–
BTFSC f,b	Testuj bit b w bajcie f, jeżeli wyzerowany to pomiń następny rozkaz	jeśli f<b>=0, to nic nie rób podczas następnego rozkazu	–
BTFSS f,b	Testuj bit b w bajcie f, jeżeli ustawiony to pomiń następny rozkaz	jeśli f<b>=1, to nic nie rób podczas następnego rozkazu	–

b – 3-bitowy numer bitu w bajcie pod adresem f,
f – zmienna dana 7-bitowym adresem w pamięci,
f<b> – bit nr b w bajcie o adresie f.

Tabela 7.6. Rozkazy operujące na stałych

Mnemonik, argument	Opis	Funkcja	Zmieniane znaczniki
ADDLW k	Dodaj W i stałą k	W + k → W	C, DC, Z
ANDLW k	Bitowy iloczyn logiczny W i stałej k	W AND k → W	Z
IORLW k	Bitowa suma logiczna W i stałej k	W OR k → W	Z
MOVLW k	Prześlij stałą k do W	k → W	–
RETLW k	Powrót z podprogramu ze stałą k w rej. W	k → W, TOS → PC	–
SUBLW k	Odejmij W od stałej k	k - W → W	C, DC, Z
XORLW k	Bitowa suma modulo 2 na W i stałej k	W XOR k → W	Z

k – 8-bitowa stała,
PC – licznik rozkazów (ang. *program counter*),
TOS – 13-bitowy adres na wierzchołku stosu (ang. *top of stack*)

Znaczniki w rej. STATUS:
C – przeniesienie,
DC – przeniesienie połówkowe,
Z – zero.

Tabela 7.7. Rozkazy sterujące

Mnemonik, argument	Opis	Funkcja	Zmieniane znaczniki
CALL a	Wywołaj podprogram	PC + 1 → TOS, a → PC<10:0> PCLATH<4:3> → PC<12:11>	–
CLRWDT	Wyzeruj licznik „Watchdoga”	0 → WDT	$\overline{TO}$, $\overline{PD}$
GOTO adr	Skok bezwarunkowy	a → PC<10:0> PCLATH<4:3> → PC<12:11>	–
RETFIE	Powrót z procedury obsługi przerwania	TOS → PC, 1 → GIE	–
RETLW k	Powrót z podprogramu ze stałą k w rej. W	k → W, TOS → PC	–
RETURN	Powrót z podprogramu	TOS → PC	–
SLEEP	Przejdź w tryb uśpienia	0 → WDT, zatrzymanie oscylatora	$\overline{TO}$, $\overline{PD}$

RETLW – ten rozkaz uwzględniono także w grupie rozkazów operujących na stałych,

k – 8-bitowa stała, GIE – bit zezwolenia na przerwania,

PC – licznik rozkazów (ang. *program counter*),

TOS – 13-bitowy adres na wierzchołku stosu (ang. *top of stack*)

Tabela 7.8. Specjalne rozkazy sterujące

Mnemonik, argument	Opis	Funkcja	Zmieniane znaczniki
OPTION	Zapisz rejestr OPTION_REG	W → OPTION_REG	–
TRIS r	Zapisz jeden z rejestrów TRISr	W → TRISr (rejestr nr r w banku 1)	–

r – 3-bitowy adres rejestru w 1 banku pamięci,
- r = 5: rejestr TRISA
- r = 6: rejestr TRISB
- r = 7: rejestr TRISC (o ile istnieje port wej./wyj. C)

Rozkazy OPTION i TRIS gwarantują zapis do odpowiedniego rejestru niezależnie od aktualnie wybranego banku pamięci.

Uwaga:
rozkazy OPTION i TRIS zostały uznane przez producenta za przestarzałe i mogą zostać usunięte z listy rozkazów w nowo opracowanych mikrokontrolerach.
Alternatywnie rejestry OPTION_REG i TRISx można zapisywać instrukcją MOVWF i odczytywać instrukcją MOVF po odpowiednim ustawieniu banku pamięci, np.:

```
banksel  TRISB
MOVWF    TRISB
```

Znaczniki wyników operacji w rejestrze STATUS

Rejestr STATUS zawiera znaczniki wyniku operacji w jednostce arytmetyczno-logicznej (ALU), bity wyboru banku pamięci danych oraz bity przyczyny (re)startu mikrokontrolera.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

R = bit do odczytu; W = bit do zapisu; 0, -1, -x – wartość po włączeniu zasilania.

Wybrane najczęściej używane bity:

bit 0 **C** (Carry): bit przeniesiena/pożyczki w operacjach arytmetycznych (instrukcje ADDLW, ADDWF, SUBLW, SUBWF)
1 = nastąpiło przeniesienie z najbardziej znaczącego bitu wyniku,
0 = nie ma przeniesienia z najbardziej znaczącego bitu wyniku.
Ponadto **C** jest 9-tym bitem w przesunięciach cyklicznych (RRF, RLF).

bit 1 **DC** (Digit Carry): bit przeniesienia/pożyczki z dolnej tetrady (połówki bajtu) używany w operacjach na kodzie BCD
1 = nastąpiło przeniesienie z najbardziej znaczącego bitu dolnej tetrady,
0 = nie ma przeniesienia z najbardziej znaczącego bitu dolnej tetrady.

bit 2 **Z** (Zero): bit zerowego wyniku operacji arytmetyczno-logicznych
1 = rezultat operacji jest zerowy,
0 = rezultat operacji jest różny od zera.

Instrukcje skoków warunkowych

W mikrokontrolerach z rodziny PIC Mid-range brak operacji skoków warunkowych pod dowolny adres; zaimplementowano jedynie cztery operacje warunkowego pominięcia następnej instrukcji w zależności od stanu bitu lub bajtu:

BTFSC f, b – testuj bit b w bajcie f, jeżeli wyzerowany to pomiń następny rozkaz,
BTFSS f, b – testuj bit b w bajcie f, jeżeli ustawiony to pomiń następny rozkaz,
DECFSZ f, d – zmniejsz f o 1, pomiń następny rozkaz jeśli wynik = 0,
INCFSZ f, d – zwiększ f o 1, pomiń następny rozkaz jeśli wynik = 0, d – gdzie zapisać wynik.

Operacje skoków warunkowych zależnych od wyniku operacji arytmetyczno-logicznej realizuje się jako instrukcje złożone z:
1) testowania wybranych znaczników rejestru STATUS i warunkowego pominięcia instrukcji,
2) skoku bezwarunkowego GOTO adres.

Przykład skoku warunkowego gdy poprzednia operacja dała wynik ≠ 0:

```
. . . . .          ;  jakaś operacja arytm.-logiczna
btfss  3, 2        ; testuj bit nr 2 (znacznik Z) pod adresem 3 (rej. STATUS)
goto   etykieta    ; skok jeżeli Z=0; rozkaz pominięty gdy Z=1
```

Pliki nagłówkowe ułatwiają zapis, np.:

```
#include p16f819.inc
. . . . .          ;  jakaś operacja arytm.-logiczna
btfss  STATUS, Z   ; testuj znacznik Z w rej. STATUS
goto   etykieta    ; skok jeżeli Z=0; rozkaz pominięty gdy Z=1
```

Pętla o zadanej liczbie powtórzeń

W mikrokontrolerach z rodziny PIC Midrange brak specjalnych instrukcji do realizacji pętli. Skutek analogiczny do instrukcji LOOP znanej z procesorów rodziny *x*86 osiąga się przez złożenie instrukcji dekrementacji i warunkowego pominięcia kolejnej instrukcji skoku bezwarunkowego.

Przykład:

```
  movlw  80h              ; zapisz stałą 80h (dziesiętnie 128) do rej. W
  movwf  licznik          ; zapisz wartość rej. W do licznika pętli

poczatek_petli
  . . .
  decfsz licznik, f       ; zmniejsz o 1 wartość licznika pętli
  goto poczatek_petli     ; wykona skok gdy wynik różny od zera; pomiń gdy Z=1
```

Analogiczna pętla w języku C:

```
licznik = 0x80;

do{
  . . .
}while(--licznik);
```

Dodawanie z uwzględnieniem przeniesienia

W mikrokontrolerach z rodziny PIC Midrange brak specjalnych operacji do wykonywania dodawania liczb wielobajtowych z uwzględnieniem przeniesienia arytmetycznego z operacji na młodszym bajcie. Warunkową poprawkę wyniku o +1 w zależności od znacznika przeniesienia wykonuje się jako operację złożoną z:

1) testowania znacznika C (albo DC dla operacji w kodzie BCD) w rejestrze STATUS
   i warunkowego pominięcia następnej instrukcji,
2) instrukcji inkrementacji.

Przykład: operacja a += b na liczbach 16-bitowych

```
movf  b_lo, W          ; pobierz mniej znaczący bajt liczby b do rej. W
addwf a_lo, f          ; zwiększ mniej znaczący bajt liczby a o wartość rej. W

movf b_hi, W           ; pobierz bardziej znaczący bajt liczby b do rej. W
btfsc STATUS, C        ; jeśli instrukcja addwf ustawiła znacznik przeniesienia C,
incf  b_hi, W          ; to dodaj poprawkę +1 do b, wynik zapisz do rej. W

addwf a_hi, f          ; zwiększ bardziej znaczący bajt liczby a o wartość rej. W
```

Oznaczenia:

a_lo, b_lo - mniej znaczące bajty liczb a oraz b,
a_hi, b_hi - bardziej znaczące bajty liczb a oraz b.

Odejmowanie z uwzględnieniem przeniesienia

W mikrokontrolerach z rodziny PIC Midrange brak specjalnych operacji do wykonywania odejmowania liczb wielobajtowych z uwzględnieniem pożyczki arytmetycznej z wartości bardziej znaczącego bajtu.

Uwaga: instrukcje SUBWF oraz SUBLW ustawiają znaczniki przeniesienia C=1 oraz DC=1 wtedy, gdy NIE występuje pożyczka. Konwencja ta jest odmienna od powszechnie przyjętej w innych procesorach, np. w instrukcjach SUB i SBB procesorów z rodziny *x*86.

Przykład: operacja a -= b na liczbach 16-bitowych

```
movf  b_lo, W          ; pobierz mniej znaczący bajt liczby b do rej. W
subwf a_lo, f          ; zmniejsz mniej znaczący bajt liczby a o wartość rej. W

movf b_hi, W           ; pobierz bardziej znaczący bajt liczby b do rej. W
btfss STATUS, C        ; jeśli instrukcja subwf nie ustawiła znacznika C,
incf  b_hi, W          ; to dodaj poprawkę +1 do b, wynik zapisz do rej. W

subwf a_hi, f          ; zmniejsz bardziej znaczący bajt liczby a o wartość rej. W
```

Oznaczenia:

a_lo, b_lo - mniej znaczące bajty liczb a oraz b,
a_hi, b_hi - bardziej znaczące bajty liczb a oraz b.

7.4. Organizacja pamięci

7.4.1. Organizacja pamięci danych

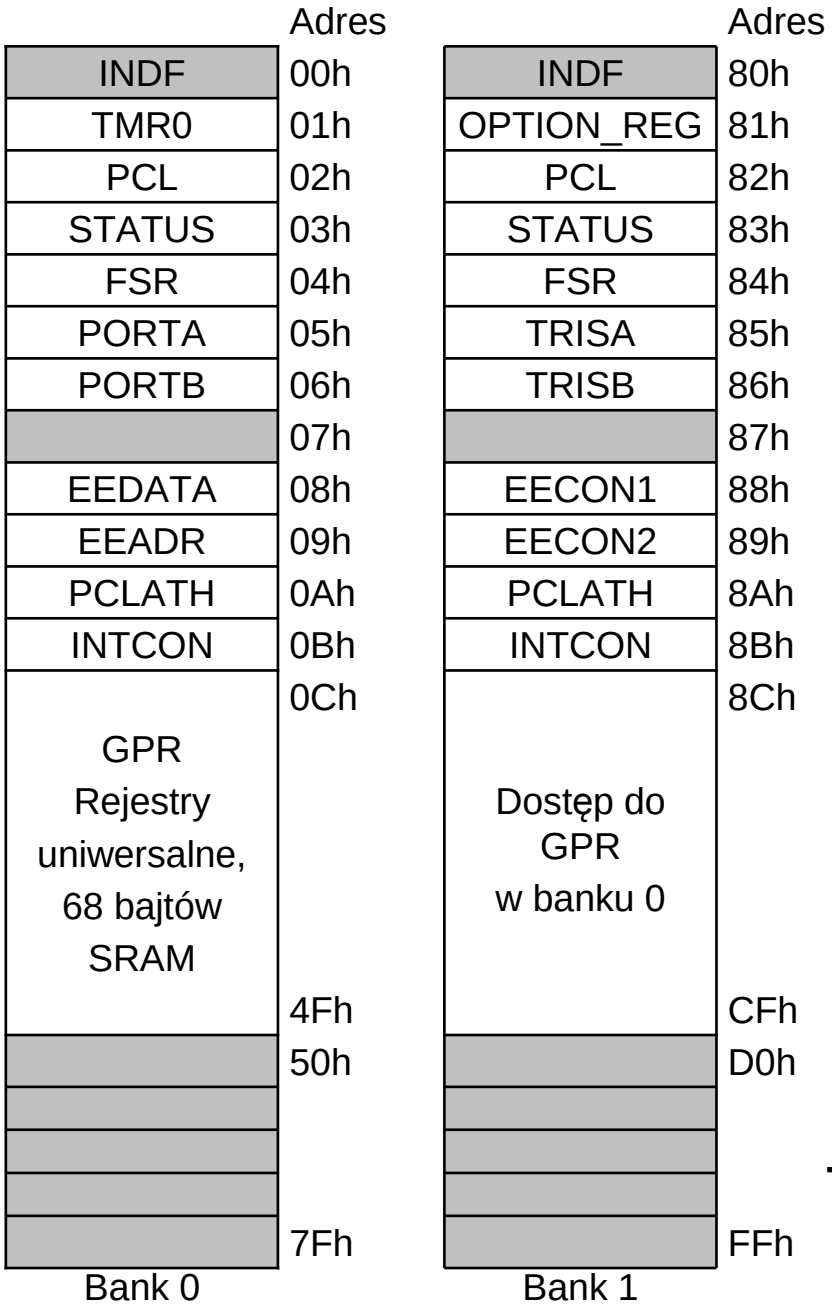
Przestrzeń adresowa pamięci danych w mikrokontrolerach PIC obejmuje:

- obszar rejestrów uniwersalnych GPR (ang. *General Purpose Registers*)
  - do wykorzystania jako pamięć danych RAM; te rejestry nie są inicjalizowane podczas resetu po włączeniu zasilania i pozostają niezmienione po innych rodzajach resetu,
- obszar rejestrów specjalnych SFR (ang. *Special Function Registers*)
  - rejestry kontrolujące pracę jednostki centralnej (CPU), np. licznik rozkazów,
  - rejestry urządzeń peryferyjnych, np. uniwersalnych portów wej./wyj.

W 8-bitowych mikrokontrolerach PIC o architekturze Midrange pamięć danych (zarówno dla GPR jak i SFR) jest podzielona na banki:

- rozmiar jednego banku wynosi 128B,
- liczba banków wynosi 2 albo 4 w zależności od modelu mikrokontrolera,
- przynajmniej 16B w każdym banku to obszar wspólnych GPR,
- aktywny bank danych wybiera się przy użyciu bitów 7…5 w rejestru **STATUS**.

Mapy pamięci



Uwaga:
Mapy pamięci poszczególnych modeli mikrokontrolerów różnią się między sobą. Odpowiednią mapę pamięci należy wyszukać w dokumentacji (ang. *Data Sheet*) dedykowanej do wybranego modelu mikrokontrolera na stronie internetowej

www.microchip.com

Oznaczenia:

	niezaimplementowane komórki, odczytywane jako 0.

INDF – nie jest fizycznym rejestrem.
Adres 0 jest używany do oznaczenia w kodzie instrukcji adresowania pośredniego.

Tabela 7.9. Przykładowa mapa pamięci mikrokontrolera PIC16F84A.

Rejestr STATUS

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

R = bit do odczytu; W = bit do zapisu; 0, -1, -x – wartość po włączeniu zasilania.

bit 7 **IRP**: numer banku pamięci danych dla adresowania pośredniego
0 = banki 0 i 1 (adresy 00h…FFh),
1 = banki 2 i 3 (adresy 100h – 1FFh).

bity 6-5 **RP<1:0>**: numer banku pamięci danych dla adresowania bezpośredniego
00 = bank 0 (adresy 00h…7Fh), 01 = bank 1 (adresy 80h…FFh),
10 = bank 2 (adresy 100h…17h), 11 = bank 3 (adresy 180h…1FFh).

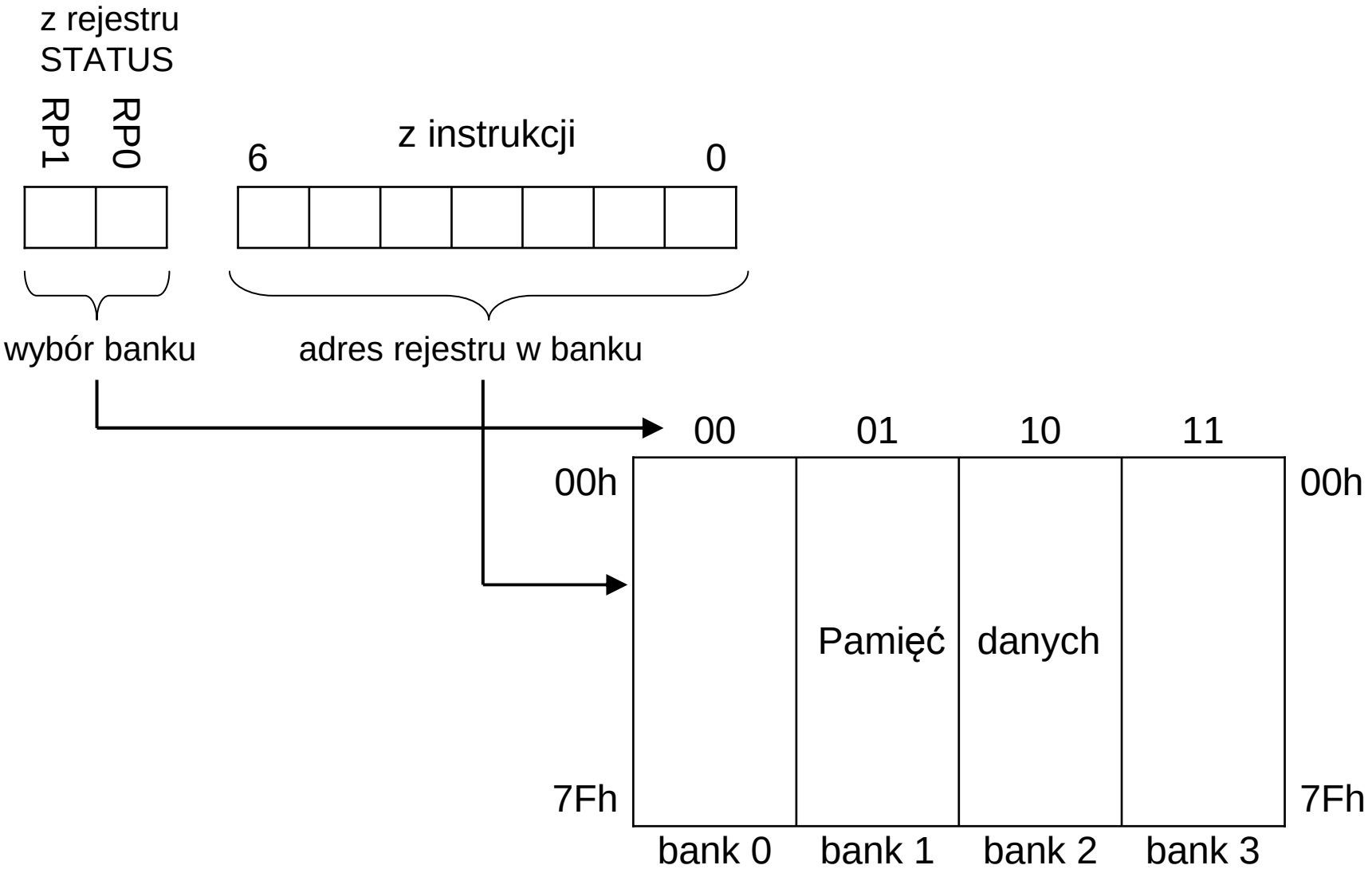
bit 4 $\overline{TO}$ (Time-out): znacznik przepełnienia licznika WDT (Watchdog Timer)
1 = po włączeniu zasilania, wykonaniu instrukcji CLRWDT lub SLEEP,
0 = po przepełnieniu się licznika WDT.

bit 3 $\overline{PD}$ (Power-down): znacznik uśpienia mikrokontrolera
1 = po włączeniu zasilania lub wykonaniu instrukcji CLRWDT,
0 = po wykonaniu instrukcji SLEEP.

Znaczniki Z, DC oraz C omówiono w rozdziale 7.3. „Rozkazy mikrokontrolerów PIC”.

Tryby adresowania pamięci danych:

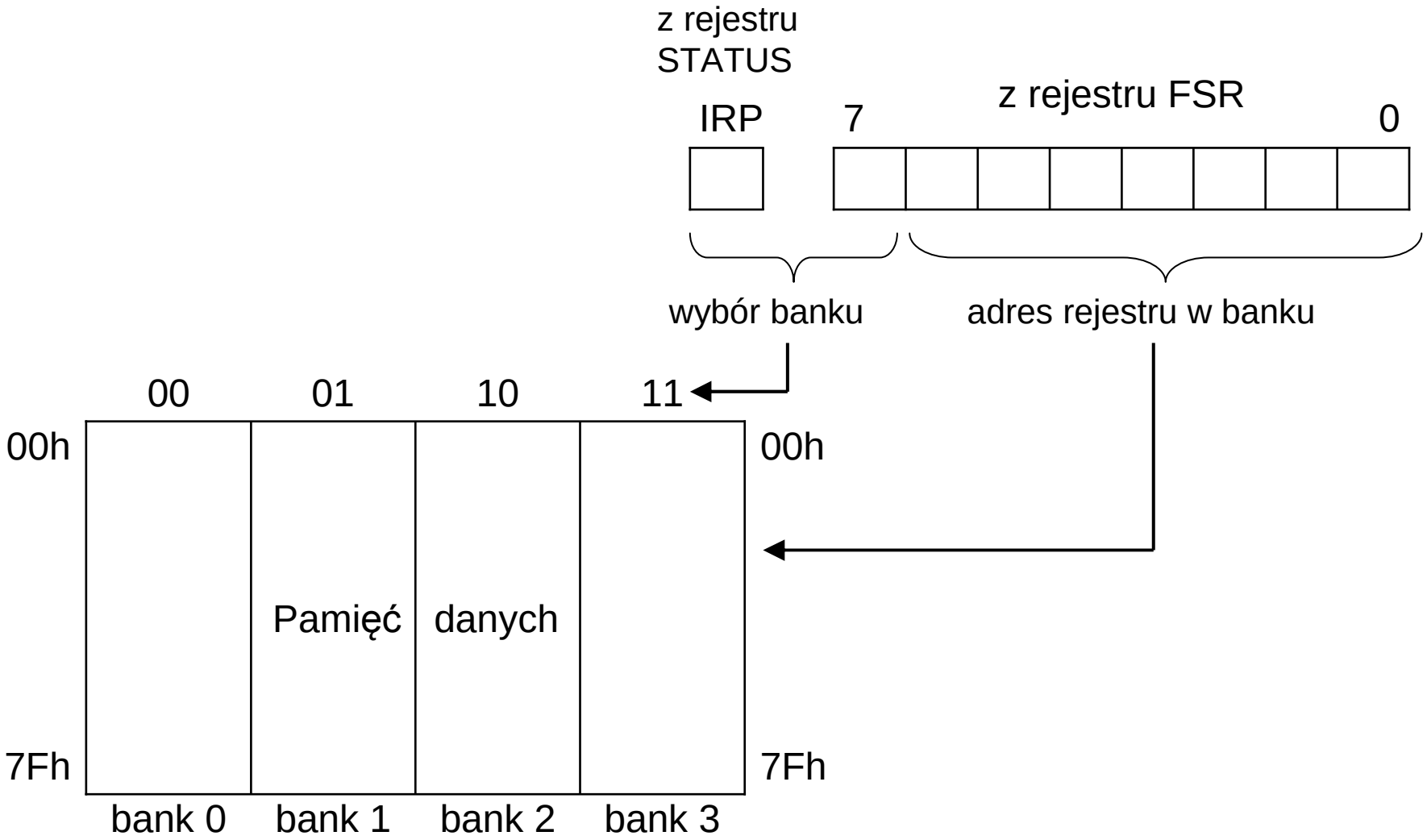
1) adresowanie bezpośrednie, tzn. przez ustalony adres zapisany w instrukcji,
2) adresowanie pośrednie - przez wskaźnik w rejestrze **FSR** (adres 04h w każdym banku).



Rys. 7.7. Adresowanie bezpośrednie pamięci danych.

Adresowanie pośrednie danych jest potrzebne m.in. do:

- operacji na komórkach w tablicach danych,
- operacji na zmiennych o nieustalonym adresie i dostępnych przez wskaźnik.

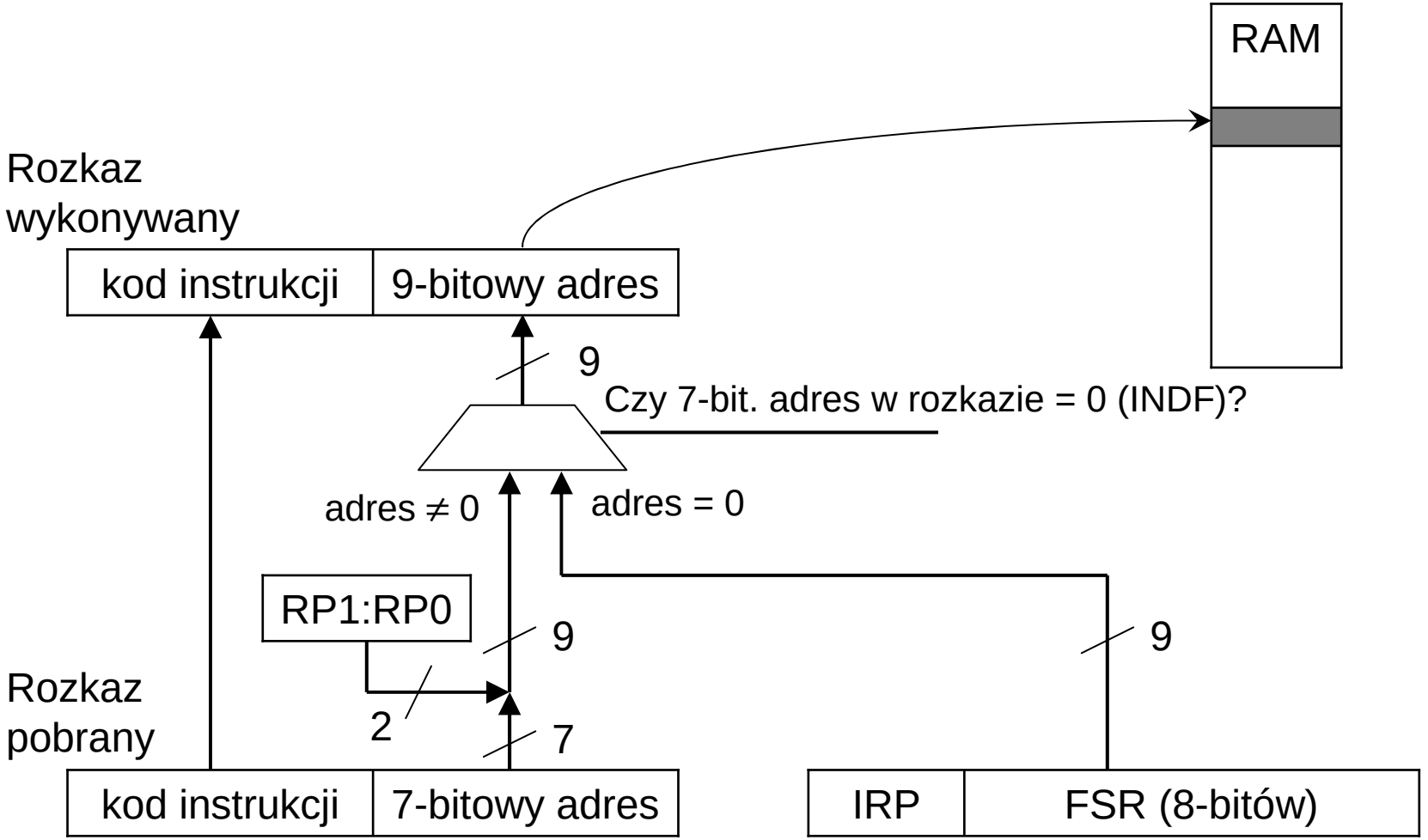


Rys. 7.8. Adresowanie pośrednie pamięci danych.

Wybór trybu adresowania bezpośredniego albo pośredniego odbywa się poprzez wartość adresu zapisanego w instrukcji:

- adres w instrukcji ≠ 0: adresowanie bezpośrednie adresem z instrukcji,
- adres w instrukcji = 0: adresowanie pośrednie przez wskaźnik w rej. FSR.

Komórka o adresie 0 nie istnieje w żadnym banku pamięci.



Rys. 7.9. Wybór trybu adresowania pamięci danych.

Przełączanie banków pamięci w asemblerze MPASM

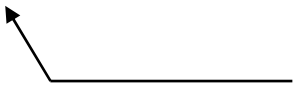
W asemblerze MPASM dostępne są predefiniowane makrodefinicje **banksel** oraz **bankisel** ułatwiające programowanie kodu przełączającego banki pamięci dla danych.

Adresowanie bezpośrednie

Składnia: banksel adres_zmiennej

Opis: Ustawia aktywny bank dla adresowania bezpośredniego, który zawiera podany 9-bajtowy adres (nie numer banku!).
Ta komenda rozwijana jest przez asembler jako instrukcje:

```
bcf / bsf  STATUS, 5
bcf / bsf  STATUS, 6
```



Stała o wartości 4 oznaczająca 9-bajtowy adres rejestru

Adresowanie pośrednie

Składnia: bankisel adres_zmiennej

Opis: Ustawia aktywne banki 0-1 albo 2-3 dla adresowania pośredniego.
Ta komenda rozwijana jest przez asembler jako instrukcja:

```
bcf / bsf  STATUS, 7
```

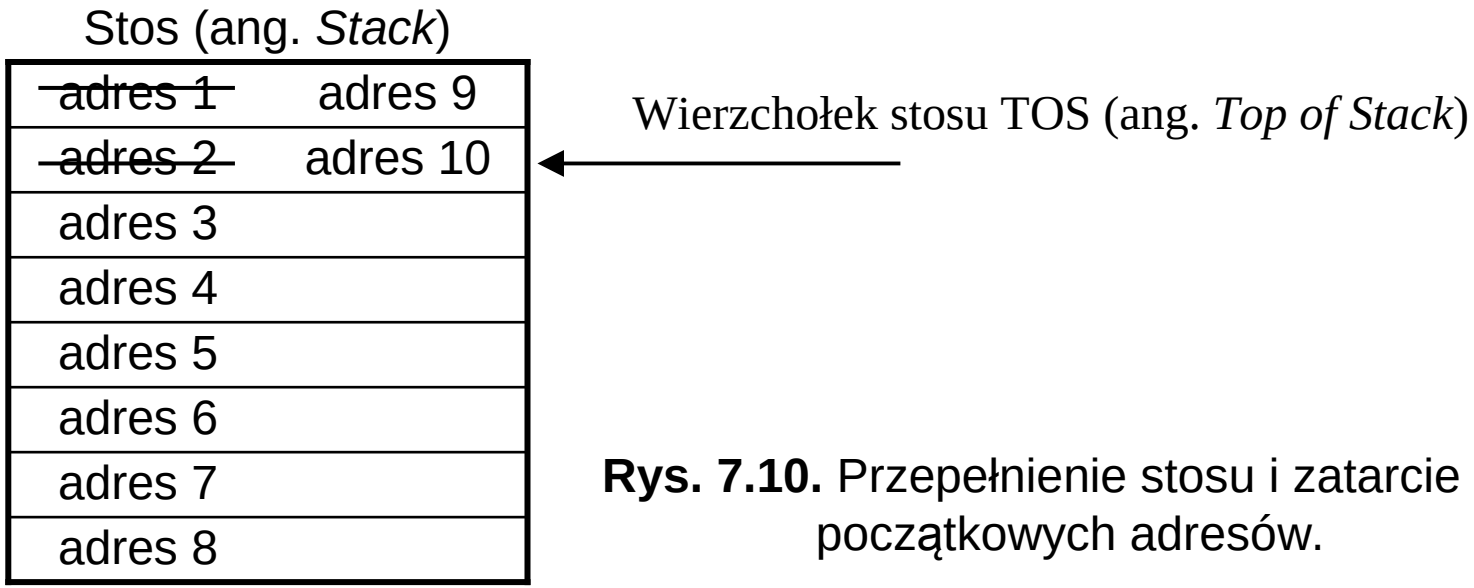
Przykłady bezpośredniego i pośredniego adresowania danych

```
#include  p16f819.inc        ; włącz plik nagłówkowy do programu
; plik nagłówkowy definiuje m.in. stałe: INDF=0  oraz   FSR=4

; definicje adresów zmiennych w GPR
zmienna1  equ  0x20
zmienna2  equ  0x21

  ; Adresowanie bezpośrednie
  banksel zmienna1          ; wybór banku 0 dla adresowania bezpośredniego
  clrf        zmienna1      ; wyzeruj zmienna1
  movlw    0x33
  movwf   zmienna2          ; zapisz 0x33 (0x - kod szestnastkowy) z rej. W do zmienna2

  ; Adresowanie pośrednie
  bankisel zmienna1         ; wybór banków 0-1 dla adresowania pośredniego
  movlw    zmienna1         ; wskaźnik do zmienna1 umieść w rej. W
  movwf    FSR              ; przepisz wskaźnik z rej. W  do rej. FSR
  clfr        INDF          ; wyzeruj zmienną daną przez wskaźnik w rej. FSR
  incf        FSR, f        ; wskaźnik w FSR += 1 (wskaźnik do nast. zmiennej)
  movlw   0x33
  movwf   INDF              ; zapisz 0x33 z rej. W do zmienna2 (przez wskaźnik)
  . . .
```

7.4.2. Stos sprzętowy

Stos sprzętowy w architekturze Midrange jest maksymalnie uproszczony:

- dane na stosie ani wskaźnik stosu (TOS) nie są dostępne w przestrzeni adresowej,
- stos sprzętowy złożony jest z 8 rejestrów 13-bitowych i może być użyty wyłącznie do przechowywania adresów powrotnych z procedur (stanów rej. PC po powrocie),
- zapis adresu na stos możliwy jest tylko poprzez instrukcję CALL lub wywołanie procedury obsługi przerwania,
- zdjęcie adresu ze stosu jest możliwe tylko poprzez wywołanie instrukcji powrotu z procedury RETURN, RETLW oraz RETFIE (powrót z obsługi przerwania),
- gdy na stosie jest już 8 adresów, następny zapis zatrze pierwszy adres – brak sprzętowych mechanizmów wykrywania przepełnienia stosu.



Rys. 7.10. Przepełnienie stosu i zatarcie początkowych adresów.

7.4.3. Organizacja pamięci programu

W 8-bitowych mikrokontrolerach PIC o architekturze Midrange pamięć programu jest adresowana następująco:

- 13-bitowy licznik rozkazów PC (ang. *Program Counter*) adresuje do 8K słów (14KB),
- rozmiar jednego banku programu przy adresowaniu bezpośrednim wynosi 2K słów 14-bitowych (w kodzie instrukcji skoków GOTO i CALL znajduje się 11 bitów adresu),
- liczba banków programu wynosi 1, 2 albo 4 w zależności od modelu mikrokontrolera,
- brak możliwości adresowania pośredniego w instrukcjach GOTO i CALL,
- wykonywanie programu po każdym rodzaju resetu zaczyna się od adresu PC=0,
- jeżeli są używane przerwania, to procedura obsługi przerwań musi zaczynać się pod ustalonym adresem 4.

Tryby adresowania skoków w pamięci programu

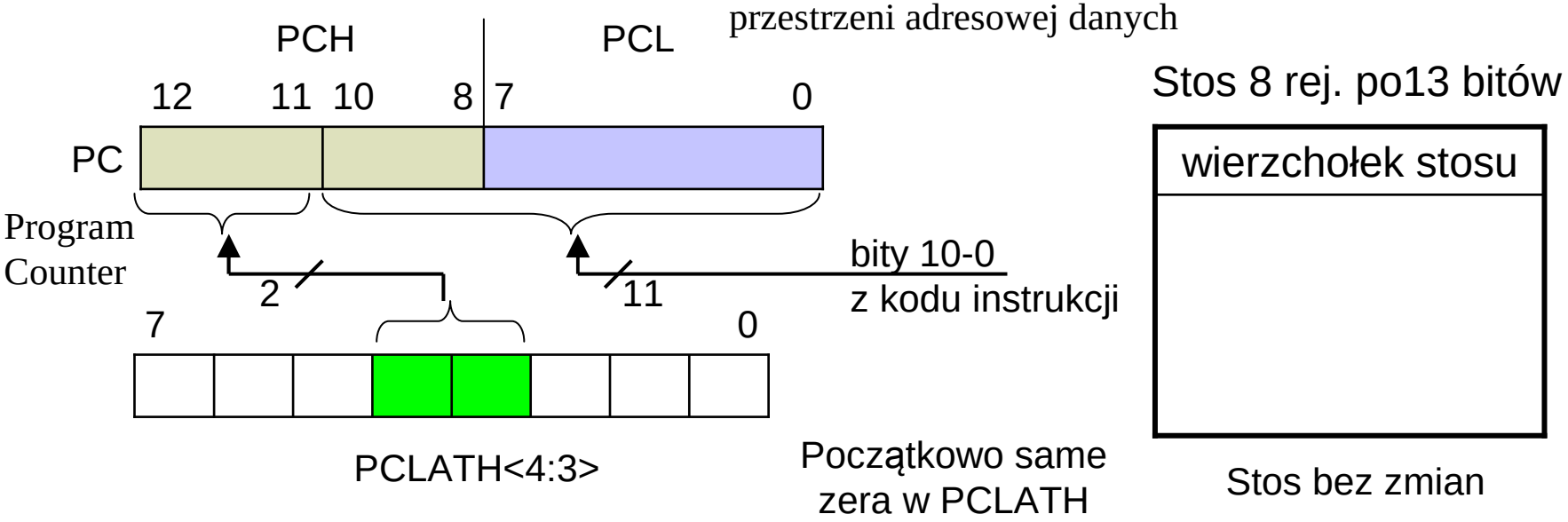
1). Adresowanie bezpośrednie (bliskie w zakresie 2K słów) – przez 11-bitowy adres w kodzie instrukcji GOTO oraz CALL.

2). Adresowanie pośrednie (bliskie w zakresie 256 słów) – poprzez zapis 8-bitowego wskaźnika bezpośrednio do mniej znaczącego bajtu licznika programu PC, który jest dostępny w przestrzeni adresowej danych jako rejestr PCL (adres 2 w każdym banku).

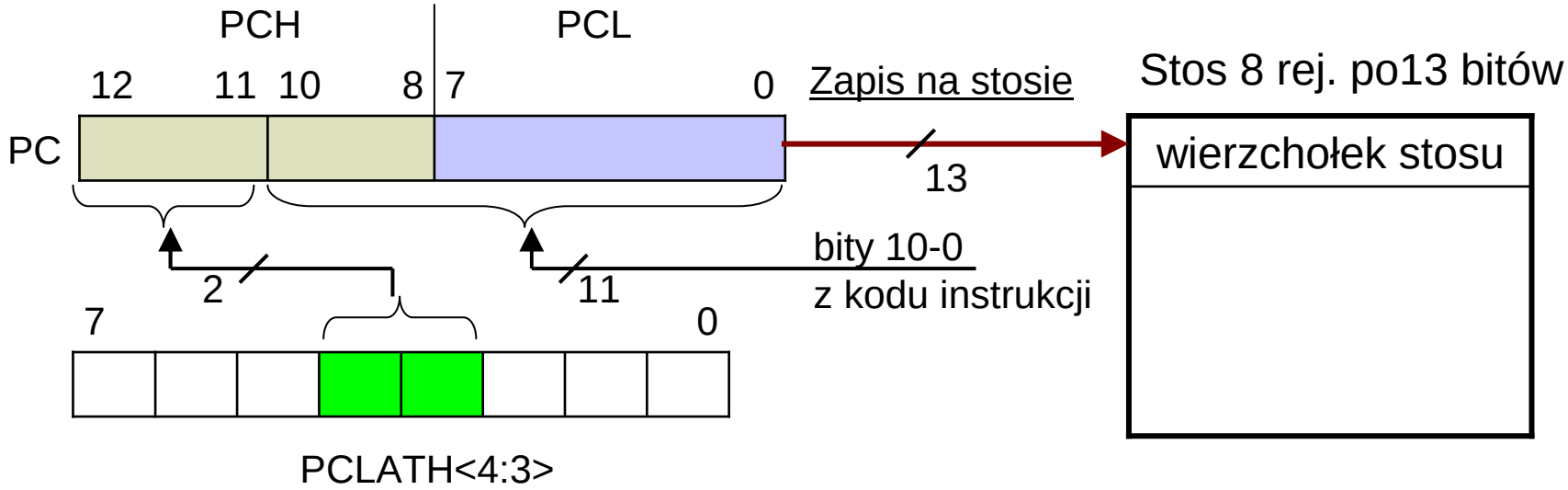
Skoki dalekie: skok bliski należy poprzedzić załadowaniem bardziej znaczących bitów adresu do rejestru zatrzaskowego PCLATH (adres 0Ah w każdym banku). Bity te zostaną przepisane z PCLATH do PC dopiero podczas najbliższego skoku lub zapisu do PCL.

Instrukcja GOTO (skok)

PCL - część PC dostępna bezpośrednio w przestrzeni adresowej danych

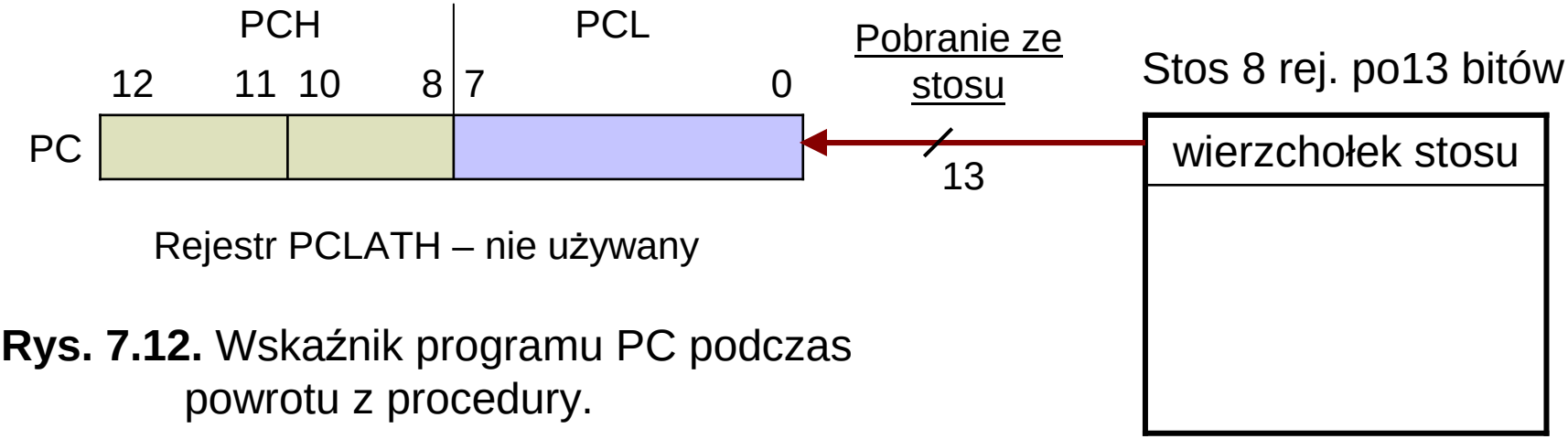


Instrukcja CALL (skok z adresem powrotu na stosie)



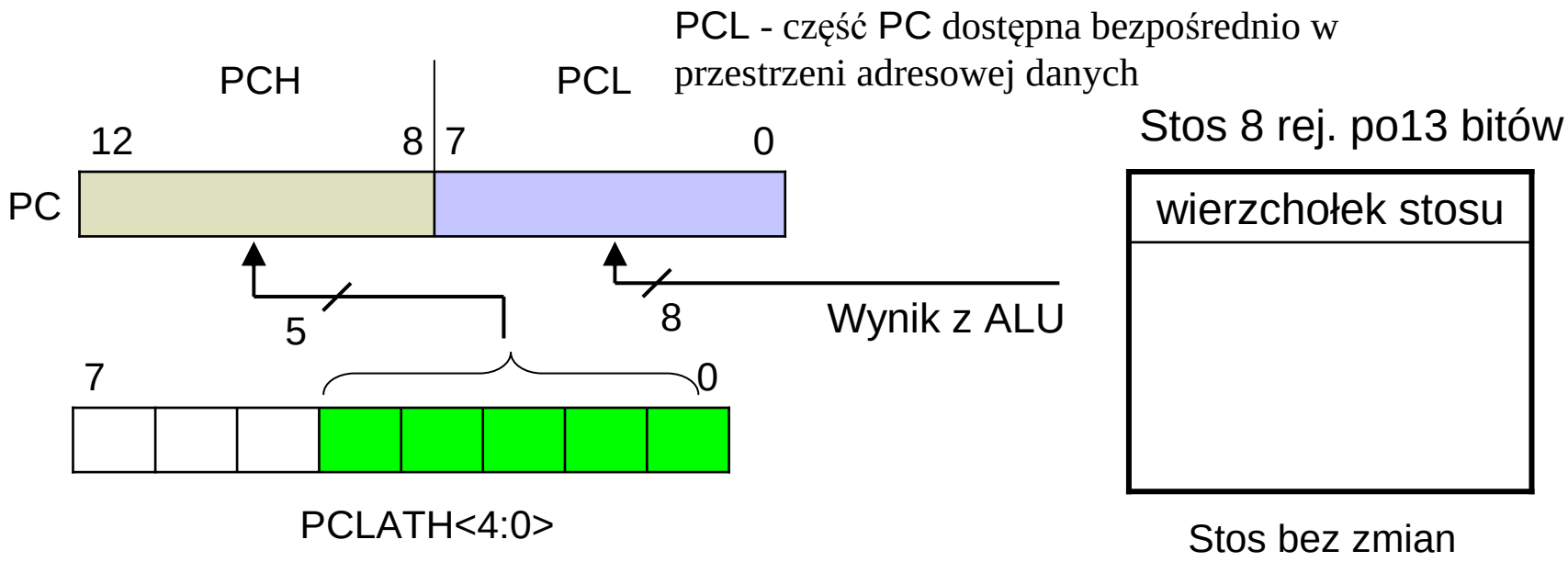
Rys. 7.11. Wskaźnik programu PC podczas skoków pod adres bezpośredni.

Instrukcje RETURN, RETLW, RETFIE (powrót z procedury)



Rys. 7.12. Wskaźnik programu PC podczas powrotu z procedury.

Instrukcje z zapisem do rejestru PCL (skoki przez wskaźnik oraz wyliczane)



Rys. 7.13. Wskaźnik programu PC podczas zapisu do rej. PCL.

Bezpośrednie i pośrednie adresowanie pamięci programu na przykładzie odczytu tablicy danych umieszczonej w kodzie programu

```
#include  p16f819.inc        ; włącz plik nagłówkowy do programu
; plik nagłówkowy definiuje m.in. stałe: PCLATH  oraz   PCL

; definicje adresów zmiennych w GPR
indeks_tabl   equ 0x20

   movlw  2
   movwf  indeks_tabl        ; przechowaj indeks =2 komórki tablicy do odczytania
   call       moja_tablica   ; wywołanie procedury pod adresem bezpośrednim
   . . .                     ; powrót z odczytaną wartością 0x81 w rej. W


   org   0x100               ; zacznij procedurę od adresu 0x100
moja_tablica                 ; procedura odczytu stałej z tablicy instrukcji retlw
   movlw HIGH(moja_tablica)  ; bardziej znaczący bajt adresu umieść w rej. W
   movwf PCLATH              ; ustaw rej. PCLATH odpowiednio do adresu tej procedury
   movf    indeks_tabl, W
   addwf  PCL, f             ; skok do wyliczonej instrukcji retlw
   retlw    0x07             ; komórka tablicy o indeksie 0
   retlw    0x88             ; komórka tablicy o indeksie 1
   retlw    0x81             ; komórka tablicy o indeksie 2
   retlw    0x32             ; komórka tablicy o indeksie 3
```

7.4.4. Pamięć konfiguracyjna

W przestrzeni adresowej programu począwszy od adresu 2000h umieszczono 14-bitowe słowa, które leżą powyżej przestrzeni adresowej dostępnej z programu wykonywanego wewnątrz MCU, natomiast są dostępne poprzez programator mikrokontrolerów:
2000h … 2003h – numer identyfikacyjny,
2006h – kod procesora,
2007h – rejestr konfiguracyjny; wartość tego rejestru należy określić przed zaprogramowaniem mikrokontrolera – zostanie ona zapisana z kodem programu.

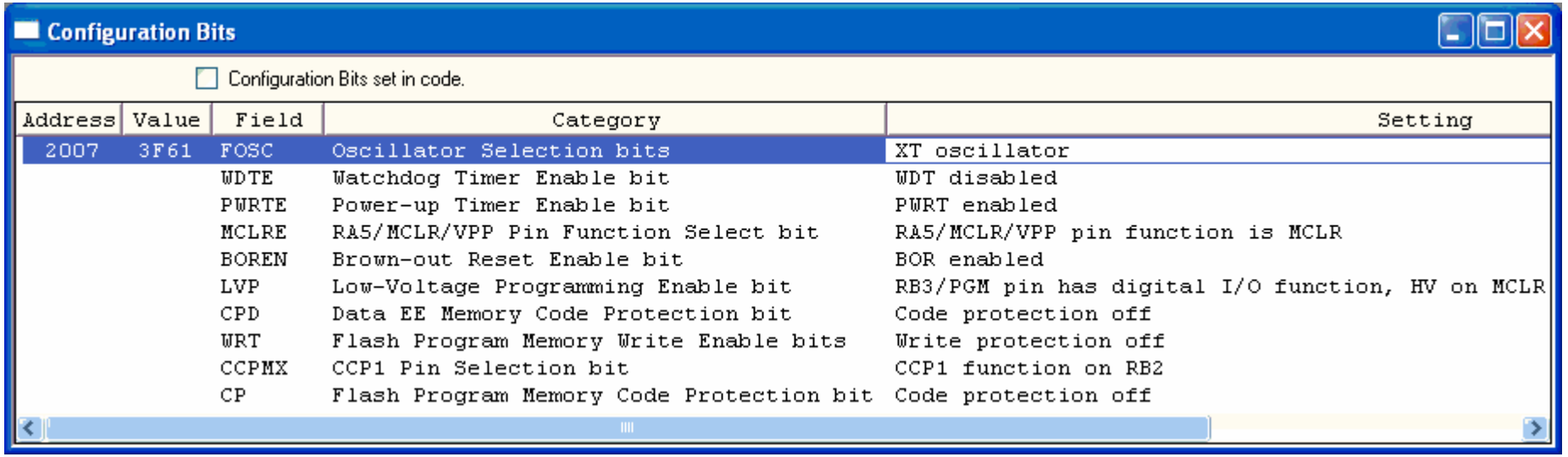
Rejestr konfiguracyjny określa:
- zabezpieczenie pamięci programu przed odczytem,
- skojarzenie funkcji CCP (ang. *Capture*, *Compare*, *PWM*) z linią RB2 albo RB3,
- włączenie debugera śledzącego pracę MCU w układzie docelowym (linie RB6 i RB7),
- zabezpieczenie pamięci programu typu Flash przed zapisem z wnętrza programu,
- zabezpieczenie pamięci danych EEPROM przed odczytem,
- programowanie niskim/wysokim napięciem (12V)
- dodatkowe opóźnienie 72ms startu po włączeniu zasilania,
- zezwolenie na różne rodzaje resetu:
  - po wykryciu spadku napięcia (ang. Brown-out Reset),
  - po przepełnieniu licznika WDT (Watchdog Timer),
  - po podaniu stanu 0 na wejście MCLR.
- wybór typu oscylatora taktującego pracę MCU.

Nie zawsze są zaimplementowane wszystkie wyliczone możliwości.

Wartość rejestru konfiguracyjnego można określić w tekście źródłowym programu w języku asembler przy użyciu komendy `__config`, np.:

```
__config  0x3F61
```

W pakiecie MPLAB IDE stan rejestru konfiguracyjnego można alternatywnie ustalić poprzez okno dialogowe Configuration Bits.



Address	Value	Field	Category	Setting
2007	3F61	FOSC	Oscillator Selection bits	XT oscillator
		WDTE	Watchdog Timer Enable bit	WDT disabled
		PWRTE	Power-up Timer Enable bit	PWRT enabled
		MCLRE	RA5/MCLR/VPP Pin Function Select bit	RA5/MCLR/VPP pin function is MCLR
		BOREN	Brown-out Reset Enable bit	BOR enabled
		LVP	Low-Voltage Programming Enable bit	RB3/PGM pin has digital I/O function, HV on MCLR
		CPD	Data EE Memory Code Protection bit	Code protection off
		WRT	Flash Program Memory Write Enable bits	Write protection off
		CCPMX	CCP1 Pin Selection bit	CCP1 function on RB2
		CP	Flash Program Memory Code Protection bit	Code protection off

Rys. 7.14. Okno dialogowe „Configuration Bits” w pakiecie MPLAB IDE v8.66. Zestaw opcji zależy od mikrokontrolera wybranego w oknie dialogowym „Select Device”. Przedstawiono przykładowe ustawienia dla mikrokontrolera PIC16F819.

Opis poszczególnych pól rejestru konfiguracyjnego podano w instrukcji do ćwiczenia laboratoryjnego E58. Dostępność poszczególnych bitów należy sprawdzać w dokumentacji dedykowanej dla wybranego modelu mikrokontrolera.

7.5. Uniwersalne porty wej./wyj.

W mikrokontrolerach PIC® z rodziny Midrange uniwersalne wejścia/wyjścia są pogrupowane w porty kontrolujące do 8 linii i zarządzane przez wspólne rejestry TRIS*x* i PORT*x*, gdzie *x*=A, B, C, D albo E. Liczba zaimplementowanych portów oraz linii wej./wyj. w poszczególnych portach jest zależna od modelu układu scalonego.

Linie skonfigurowane do pełnienia innych specjalnych zadań (np. wejścia analogowe, wejście resetu) nie mogą być jednocześnie używane jako uniwersalne cyfrowe wejścia/wyjścia.

Tabela 7.10. Liczba uniwersalnych linii wej./wyj. zaimplementowanych w wybranych mikrokontrolerach PIC z rodziny Midrange (dostępne są w Laboratorium Techniki Cyfrowej Instytutu Fizyki PŁ).

Model układu	port A	port B	port C	port D	port E
PIC16F84A	5 (a)	8	–	–	–
PIC16F819	8 (b)	8	–	–	–
PIC16F877A	6 (a)	8	8	8	3

(a) wyjście na linii RA4 jest typu „otwarty dren” w PIC16F84A i PIC16F877A.
(b) linia RA5 pracuje tylko jako wejście w PIC16F819.

Z każdym portem A, B, C, D i E związane są po dwa rejestry TRIS*x* oraz PORT*x* dostępne w przestrzeni adresowej danych, gdzie *x*=A, B, C, D albo E.

bit:	7	6	5	4	3	2	1	0
TRISA	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0

Bit nr *n* kontroluje linię wej./wyj. RA*n*

- TR*n* = 0: linia skonfigurowana jako wyjście;
  stan logiczny wyjścia zależy od bitu RA*n* w rejestrze PORTA.
- TR*n* = 1: linia skonfigurowana jako wejście;
  sterownik wyjścia jest ustawiony w stan wysokiej impedancji.

Po włączeniu zasilania i każdym rodzaju resetu wszystkie zaimplementowane bity rejestrów TRIS*x* są ustawiane w stan 1.

bit:	7	6	5	4	3	2	1	0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0

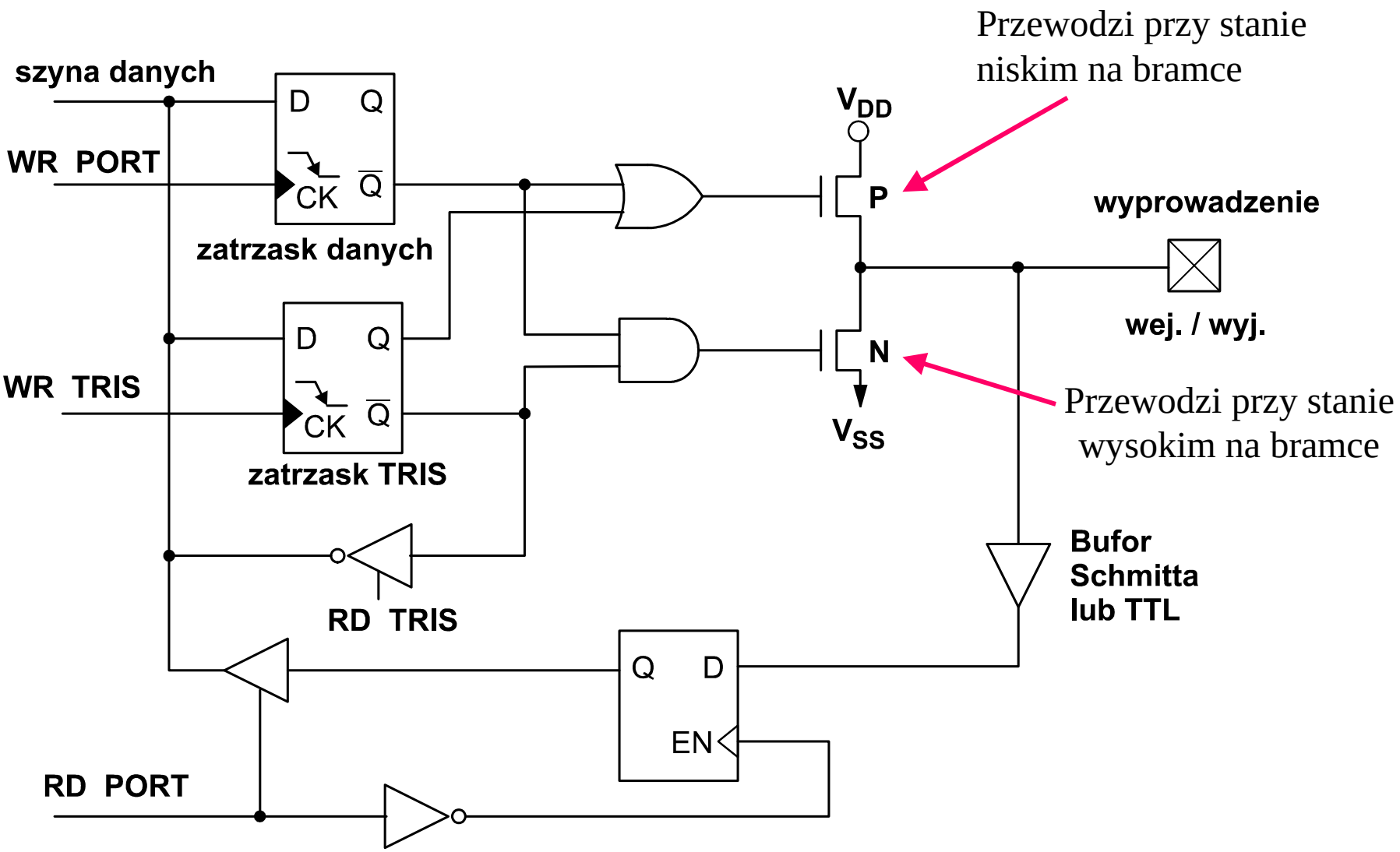
Jeżeli w rejestrze TRISA bit TR*n* = 0, to:

- RA*n* = 0: wyjście w stanie 0 (wyjście połączone z masą),
- RA*n* = 1: wyjście w stanie 1 (połączenie z +Ucc, albo stan wysokiej impedancji dla wyjść typu „otwarty dren”).

Początkowy stan rejestrów PORT*x* nie jest ujednolicony w różnych modelach MCU.

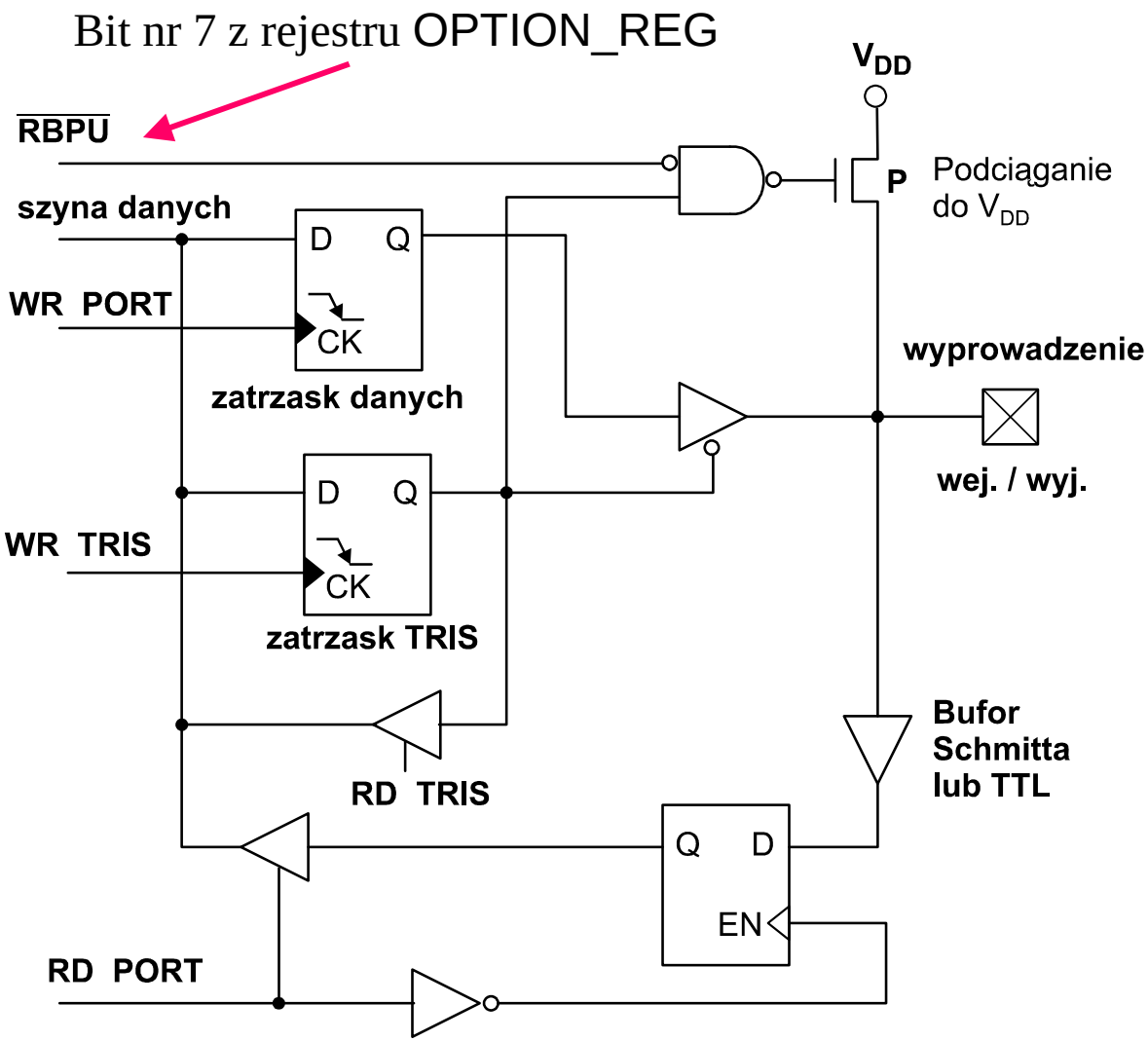
Bieżący stan linii można odczytać z rej. PORT*x*, niezależnie od stanu rej. TRIS*x*.

Typowa budowa kontrolerów linii portu A



Rys. 7.16. Uproszczony schemat blokowy uniwersalnych portów wej./wyj. RA*x*. Pominięto dodatkowe funkcje.

Typowa budowa kontrolerów linii portu B



Unikalną cechą portu B jest możliwość włączenia podciągania napięcia na linii pracującej jako wejście (bit w rej. TRIS = 0) do napięcia zasilania V_{DD} poprzez dodatkowy tranzystor pracujący jako programowany rezystor.

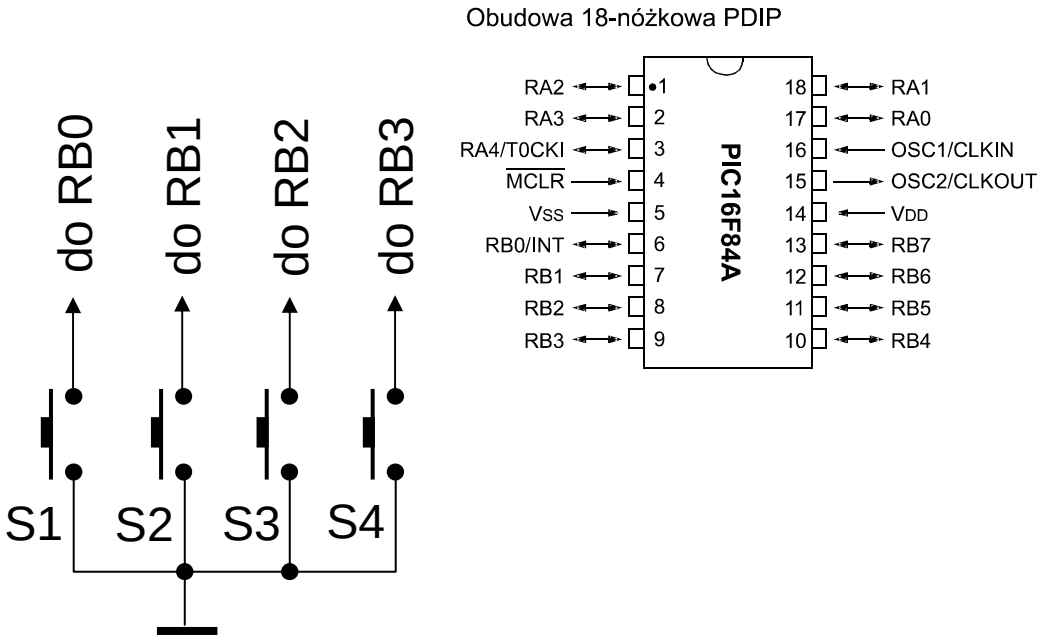
Właściwość ta umożliwia odczytywanie stanu przełączników bez dodatkowych zewnętrznych rezystorów.

Rys. 7.16. Uproszczony schemat blokowy uniwersalnych portów wej./wyj. RB*x*. Pominięto dodatkowe funkcje.

Odczyt stanu przycisków z zestawie ZL4PIC

Zestaw uruchomieniowy ZL4PIC dostępny w lab. techniki cyfrowej posiada przyciski S1, S2, S3, S4 (nie licząc resetu S5), które:

- podczas przyciskania zwierają wejście mikrokontrolera do masy – odczyt stanu 0,
- gdy przycisk jest puszczony swobodnie odczyt stanu zależy od załączenia w mikrokontrolerze rezystora podciągającego stan wejścia do napięcia zasilania:
  - gdy rezystor załączony odczyt stanu 1,
  - **gdy rezystor niezałączony nie wiadomo jaki stan zostanie odczytany!**

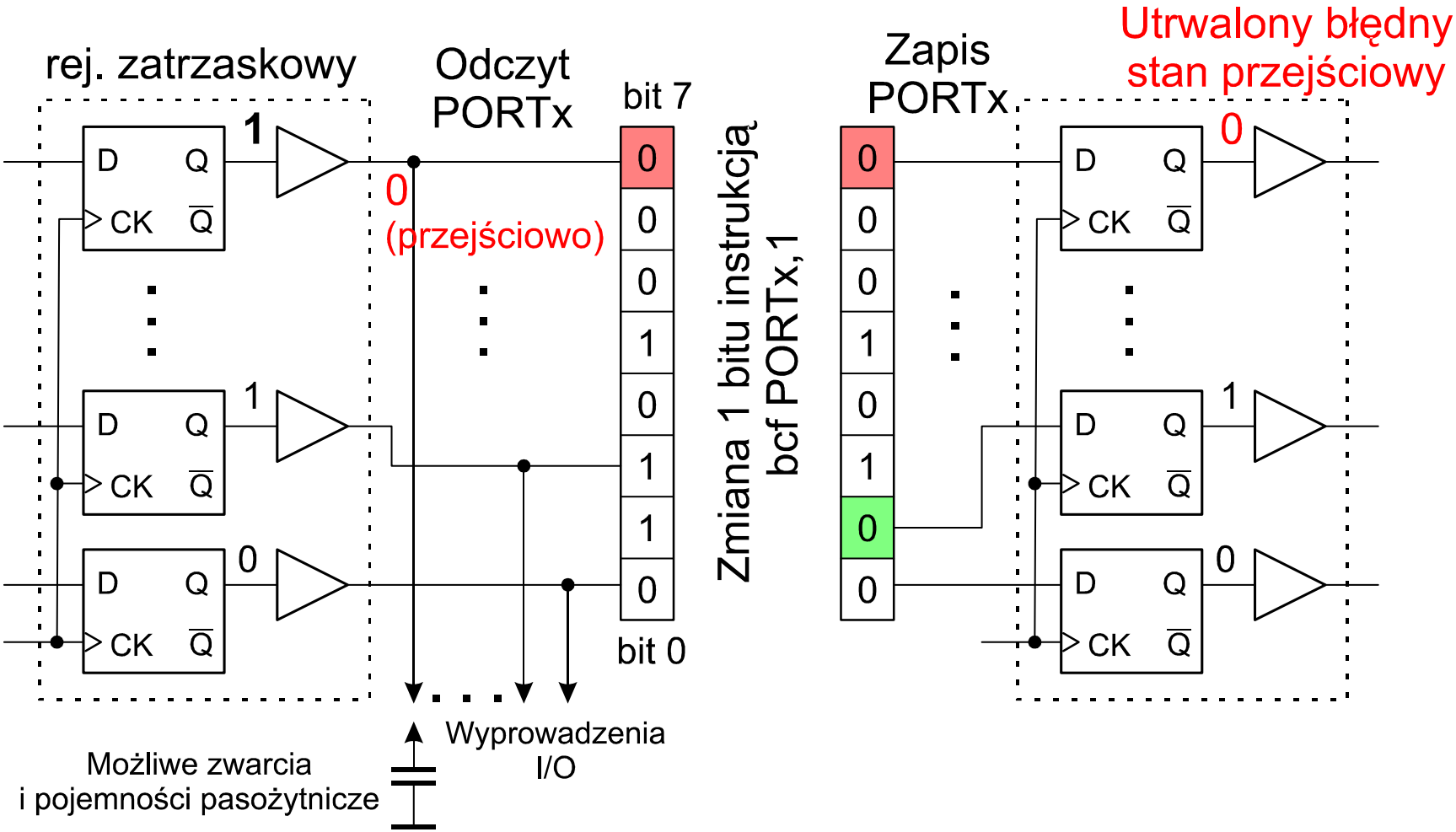


Rys. 7.17. Podłączenie przycisków w zestawie uruchomieniowym ZL4PIC.

```
// rezystory domyślnie wyłączone
banksel PORTB
btfsc PORTB, 0  // Błąd! Loteria!
goto ...

banksel OPTION_REG
bcf OPTION_REG, NOT_RBPU

banksel PORTB
btfsc PORTB, 0  // Prawidłowo
goto ...
```

Uwaga: odczyt z rejestrów PORT*x* zwraca bieżący stan linii portu, a nie stan zapisany ostatnio do rejestru zatrzaskowego. Operacje na bitach bcf / bsf przetwarzają cały bajt w cyklu odczyt-modyfikacja-zapis.

Przykład: bcf PORTx, 1 ; zamierzona zmiana tylko bitu 1



Rys. 7.18. Mechanizm utrwalania błędnych stanów przejściowych podczas bezpośredniej modyfikacji rejestrów PORTA, PORTB, … .

Przykład operacji na liniach portu A

```
#include  p16f84A.inc        ; włącz plik nagłówkowy do programu

; definicje adresów zmiennych w GPR
bufor_portA  equ  0x0C

 ; Inicjalizacja portu A
  banksel  PORTA             ; wybór banku 0
  clrf     PORTA             ; wyzeruj wyjściowe zatrzaski danych
  clrf     bufor_portA
  banksel TRISA              ; wybór banku 1
  movlw    b'11110000'
  movwf   TRISA              ; ustaw RB0…3 jako wyjścia; reszta wejścia
  banksel  PORTA             ; wybór banku 0
  . . .

 ; Wariant hazardowy – operacje wprost na PORTx
  bsf      PORTA, 1          ; zamierzone ustawienie tylko bitu 1
  bcf      PORTA, 2          ; zamierzone wyzerowanie tylko bitu 2

 ; Wariant bezpieczny – operacje buforowane w pamięci
  bsf      bufor_portA, 1
  bcf      bufor_portA, 2
  movf     bufor_portA, W
  movwf    PORTA             ; przepisz zmiany z bufora do PORTA
```

7.6. Literatura

[1] *PICmicro™ Mid-Range MCU Family Reference Manual*, Data Sheet DS33023A, Microchip Technology Inc. 1997, dostępne na stronie www.microchip.com.

[2] *PIC16F84A*, Data Sheet DS35007C, Microchip Technology Inc. 2001-2013, dostępne na stronie www.microchip.com.

[3] *PIC16F818/819*, Data Sheet DS39598F, Microchip Technology Inc. 2001-2013, dostępne na stronie www.microchip.com.

[4] *PIC16F87XA*, Data Sheet DS39582C, Microchip Technology Inc. 2001-2013, dostępne na stronie www.microchip.com.

[6] T. Jabłoński, *Mikrokontrolery PIC16F8x w praktyce*, Wydawnictwo BTC, Warszawa 2002.

[7] S. Pietraszek, *Mikroprocesory jednoukładowe PIC*, Helion, Gliwice 2002.

[8] T. Jabłoński, K. Pławsiuk, *Programowanie mikrokontrolerów PIC w języku C*, Wydawnictwo BTC, Warszawa 2005.