Lista operacji procesorów PIC18F ... K ...

Listę instrukcji przepisano z dokumentacji mikrokontrolera PIC18F14K22. Procesory PIC18 innych serii niż „K” mogą nie mieć niektórych operacji lub mieć takie, których seria „K” nie posiada.

Literka *d* w poniższym spisie oznacza *destination*, czyli wskazanie, do którego rejestru zostanie wpisany wynik operacji. Jeśli podamy w tym miejscu „w” (0) to asembler potraktuje to jako wpisanie rezultatu do rejestru WREG. Jesli podamy „f” (1) to wynik znajdzie się w rejestrze podanym w instrukcji. **Pominięcie oznacza 1 (rejestr).**

Literka *a* to operand przeznaczony na wskazanie sposobu adresowania rejestru z danymi. Jeżeli jako *a* podamy 0, adresowanie będzie uczynione z użyciem rejestru z grupy wszędzie dostępnej, tzn. niezależnie od ustawionego banku pamięci (tzw. *Access RAM*). Przy 1 będzie to adres w obrębie aktualnego banku pamięci. Niestety, nie ma tu ładnego literowego skrótu, jak w przypadku operandu *d*. **Pominięcie oznacza 0 (dostęp na zasadach *Access RAM*).**

Operacje zorientowane na bajt

Mnemonik i parametry	Wykonywana operacja	Flagi, na które ma wpływ instrukcja
ADDWF *f, d, a*	Dodaj WREG i *f*	C, DC, Z, OV, N
ADDWFC *f, d, a*	Dodaj WREG i bit przeniesienia (C) do *f*	C, DC, Z, OV, N
ANDWF *f, d, a*	Iloczyn logiczny (AND) WREG i *f*	Z, N
CLRF *f, a*	Wyzeruj *f*	Z
COMF *f, d, a*	Dopełnij binarnie *f*	Z, N
CPFSEQ *f, a*	Porównaj *f* z WREG, przeskocz jedną instrukcję gdy *f* = WREG	
CPFSGT *f, a*	Porównaj *f* z WREG, przeskocz jedną instrukcję gdy *f* > WREG	
CPFSLT *f, a*	Porównaj *f* z WREG, przeskocz jedną instrukcję gdy *f* < WREG	
DECF *f, d, a*	Zmniejsz *f* o 1	C, DC, Z, OV, N
DECFSZ *f, d, a*	Zmniejsz *f* o 1, przeskocz jedną instrukcję jeśli 0	
DCFSNZ *f, d, a*	Zmniejsz *f* o 1, przeskocz jedną instrukcję jeśli nie 0	
INCF *f, d, a*	Zwiększ *f* o 1	C, DC, Z, OV, N
INCFSZ *f, d, a*	Zwiększ *f* o 1, przeskocz jedną instrukcję jeśli 0	
INFSNZ *f, d, a*	Zwiększ *f* o 1, przeskocz jedną instrukcję jeśli nie 0	
IORWF *f, d, a*	Suma logiczna (OR) WREG i *f*	Z, N

MOVF *f, d, a*	Przesuń *f*	Z, N
MOVFF *fs, fd*	Przesuń *fs* do *fd*	
MOVWF *f, a*	Przesuń WREG do *f*	
MULWF *f, a*	Pomnóż WREG i *f*	
NEGF *f, d, a*	Zaneguj *f*	C, DC, Z, OV, N
RLCF *f, d, a*	Obróć *f* w lewo wraz z bitem przeniesienia (C)	C, Z, N
RLNCF *f, d, a*	Obróć *f* w lewo (bez użycia bitu przeniesienia)	Z, N
RRCF *f, d, a*	Obróć *f* w prawo wraz z bitem przeniesienia (C)	C, Z, N
RRNCF *f, d, a*	Obróć *f* w prawo (bez użycia bitu przeniesienia)	Z, N
SETF *f, a*	Ustaw wszystkie bity w *f*	
SUBFWB *f, d, a*	Odejmij *f* od WREG wraz z bitem pożyczki (C)	C, DC, Z, OV, N
SUBWF *f, d, a*	Odejmij WREG od *f*	C, DC, Z, OV, N
SUBWFB *f, d, a*	Odejmij WREG of *f* wraz z bitem pożyczki (C)	C, DC, Z, OV, N
SWAPF *f, d, a*	Zamień połówki w *f*	
TSTFSZ *f, a*	Sprawdź *f*, przeskocz jedną instrukcję jeśli 0	
XORWF *f, d, a*	Alternatywa wykluczająca WREG i *f*	Z, N

Operacje zorientowane na bit

BCF *f, b, a*	Wyzeruj bit nr *b* z rejestru *f*	
BSF *f, b, a*	Ustaw bit nr *b* z rejestru *f*	
BTFSC *f, b, a*	Sprawdź bit *b* w *f*, przeskocz jedną instrukcję jeśli bit jest wyzerowany	
BTFSS *f, b, a*	Sprawdź bit *b* w *f*, przeskocz jedną instrukcję jeśli bit jest ustawiony	
BTG *f, b, a*	Zmień na przeciwny bit *b* w rejestrze *f*	

Operacje kontrolne

BC *n*	Skocz do adresu *n*, jeśli jest ustawione przeniesienie (C=1). Adresowanie względne	
BN *n*	Skocz do adresu *n*, jeśli jest ustawiony bit ujemności (N=1). Adresowanie względne	
BNC *n*	Skocz do adresu *n*, jeśli jest wyzerowany bit przeniesienia (C=0). Adresowanie względne	
BNN *n*	Skocz do adresu *n*, jeśli jest wyzerowany bit negatywności (N=0). Adresowanie względne	
BNOV *n*	Skocz do adresu *n*, jeśli nie było przepełnienia	

	(OV=0)	
BNZ *n*	Skocz do adresu *n*, jeśli jest wyzerowany bit zerowości (Z=0). Adresowanie względne	
BOV *n*	Skocz do adresu *n*, jeśli było przepełnienie (OV=1). Adresowanie względne	
BRA *n*	Bezwarunkowo skocz do adresu *n*. Adresowanie względne	
BZ *n*	Skocz do adresu *n*, jeśli jest ustawiony bit zerowości (Z=1). Adresowanie względne	
CALL *k*, *s*	Wywołaj podprogram znajdujący się pod adresem *k*. Adresowanie **bezwzględne**	
CKRWDT	Wyczyść timer Watchdoga	$\overline{\text{TO}}$, $\overline{\text{PD}}$
DAW	Wyrównaj dziesiętnie WREG	C
GOTO *k*	Bezwarunkowo skocz do adresu *k*. Adresowanie **bezwzględne**	
NOP	Nie rób nic	
POP	Pobierz wierzchołek stosu powrotu	
PUSH	Złóż na wierzchołek stosu powrotu	
RCALL *n*	Wywołaj podprogram znajdujący się pod adresem *n*. Adresowanie wzgledne	
RESET	Wykonaj programowy reset urządzenia	wszystkie
RETFIE *s*	Powrót z przerwania i ponowne umożliwienie przerwań	GIE/GIEH, PEIE/GIEL
RETLW *k*	Powróć z podprogramu z podanym operandem *k*, wpisując go do WREG	
RETURN *s*	Powróć z podprogramu	
SLEEP	Przejdź w tryb uśpienia	$\overline{\text{TO}}$, $\overline{\text{PD}}$

Operacje na operandzie bezpośrednim

ADDLW *k*	Dodaj operand *k* do WREG	C, DC, Z, OV, N
ANDLW *k*	Iloczyn logiczny operandu *k* i WREG	Z, N
IORLW *k*	Suma logiczna operandu *k* i WREG	Z, N
LFSR *f*, *k*	Przesuń 12-bitowy operand *k* do jednego z trzech rejestrów adresowania pośredniego (FSR0, FSR1 lub FSR2) podanego operandem *f*	
MOVLB *k*	Przesuń 4-bitowy operand *k* do rejestru selekcji banku pamięci (BSR<3:0>)	
MOVLW *k*	Przesuń operand *k* do WREG	
MULLW *k*	Pomnóż operand *k* przez WREG	

RETLW *k*	Powróć z podprogramu z podanym operandem *k*, wpisując go do WREG	
SUBLW *k*	Odejmij WREG od operandu *k*	C, DC, Z, OV, N
XORLW *k*	Alternatywa wykluczająca (XOR) operandu *k* i WREG	Z, N

Operacje wymiany danych z pamięcią programu

TBLRD*	Odczytaj tabelę	
TBLRD*+	Odczytaj tablicę z postinkrementacją (zwiększeniem wskaźnika tebeli **po** przeczytaniu)	
TBLRD*-	Odczytaj tabelę z postdekrementacją (zmniejszeniem wskaźnika tabeli **po** przeczytaniu)	
TBLRD+*	Odczytaj tabelę z preinkrementacją (zwiększeniem wskaźnika tabeli **przed** przeczytaniem)	
TBLWT*	Zapisz tabelę	
TBLWT*+	Zapisz tabelę z postinkrementacją (zwiększeniem wskaźnika tabeli **po** przeczytaniu)	
TBLWT*-	Zapisz tabelę z postdekrementacją (zmniejszeniem wskaźnika tabeli **po** przeczytaniu)	
TBLWT+*	Zapisz tabelę z preinkrementacją (zwiększeniem wskaźnika tabeli **przed** przeczytaniem)	

Instrukcje rozszerzające podstawowy zestaw

Umożliwienie (bitami konfiguracyjnymi mikrokontrolera) wykonywania rozszerzonego zestawu instrukcji może prowadzić starsze aplikacje do błędnej pracy. Zmienia to pracę podstawowego zestawu instrukcji. Mimo to roszerzone instrukcje są tu dodane dla kompletności zestawienia. Tylko w niektórych typach programowania coś wnoszą i są wykorzystywane głównie przez języki programowania wysokiego poziomu. Kolidują z pełnym wykorzystaniem rejestrów z grupy zawsze dostępnej *Access RAM*. Więcej w dokumencie *PIC18(L)F1XK22 Data Sheet* w punkcie 24.2.3 zatytułowanym „BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE” (str. 312). Do pobrania z serwera Microchip-a

ADDFSR *f, k*	Dodaj 6-bitowy operand *k* do FSR(f)	
ADDULNK *k*	Dodaj 6-bitowy operand *k* do FSR2 i powróć z podprogramu	
CALLW	Wywołaj podprogram używając WREG. Będzie on wpisany do PCL jako najniższy bajt adresu, wyższe	

	zostaną przepisane z PCLATH i PCLATU	
MOVSF *zs, fd*	Przesuń zawartość 7-bitowo indeksowanego rejestru *zs* do w pełni adresowanego (12-bitowo) *fd.* Pełny adres rejestru *zs* jest uzyskiwany przez dodanie do *zs* do FSR2	
MOVSS *zs, zd*	Przesuń zawartość 7-bitowo indeksowanego *zs* do 7-bitowo indeksowanego *zd*. Oba adresy są konstruowane przez dodanie operandu do FSR2	
PUSHL *k*	Zachowaj 8-bitowy operand bezpośredni *k* pod adresem wskazanym przez FSR2 a następnie zmniejsz o 1 FSR2. Umożliwia to złożenie wartości na stosie programowym.	
SUBFSR *f, k*	Odejmij 6-bitowy operand bezpośredni *k* od FSR(f)	
SUBULNK *k*	Odejmij 6-bitowy operand bezpośredni od FSR2 i wróć z podprogramu	