부록 2. 개별 명령어 설명

ADDLW k Add Literal to W

• 설명 W 레지스터의 값과 상수 k의 값을 더한 뒤 그 결과를 W레지스터에 저장
• 처리 (W)+k --> (W)
• operands 0 ≤ k ≤ 255
• STATUS C,DC,Z
• 사이클 1
+ 14BIT OPCODE

MSB LSB
11 111x kkkk kkkk ; x 는 don't care

• 사용 예 ADDLW 11H (기계어코드: 3E11)

실행전	실행후
(W) = 11H	(W) =22H

결과가 0 이 아니므로 Z플래그는 0이 되고, 또 bit7과 bit3에서 자리올림이 발생하지 않았으므로 C플래그와 DC플레그는 0이 된다.

• 사용 예 ADDLW 88H (기계어코드: 3E88)

실행전	실행후
(W) = 78H	(W) =00H

결과가 0 이므로 Z플래그는 1이 되고, 또 bit7과 bit3에서 자리올림이 발생하였으므로 C플래그와 DC플레그는 1이 된다.

ADDWF f, d Add W to f

• 설명 W레지스터의 값과 f레지스터의 값을 더한 뒤 그 결과를 d로 보낸다.
(d가 '0'이면 W레지스터에 저장하고, "1"이면 f레지스터에 저장한다.)
• 처리 (W) + (f) --> (d)
• STATUS C,DC,Z
• 사이클 1
+ 14BIT OPCODE

MSB LSB

00 0111 dfff ffff

• 사용예 **ADDWF 20H,1 (기계어코드: 07A0)**

실행전	실행후
(W) = 10H (20H) = 15H	(W) = 10H (20H) = 25H

• 사용예 **ADDWF 20H,0 (기계어코드: 0720)**

실행전	실행후
(W) = 10H (20H) = 15H	(W) = 25H (20H) = 15H

ANDLW k AND Literal and W

• 설명 W레지스터의 값과 상수 k를 AND 연산한 뒤 그 결과를 W레지스터에 저장한다.

• 처리 **(W) . AND. K-->(W)**

• operands $0 \leq k \leq 255$

• STATUS **Z**

• 시이클 1

+ 14BIT OPCODE

MSB LSB

11 1001 kkkk kkkk

• 사용예 **ANDLW 0FH (기계어코드: 390F)**

실행전	실행후
(W) = 03AH	(W) = 0AH

결과적으로 실행 전 W레지스터에서 상위 4BIT는 강제로 0으로 만들고, 하위 4비트만 그대로 남게 된다. 이와 같은 작업(원하는 작업만 남기는 작업)을 “비트 마스크”라고 부른다.

ANDWF f,d AND W with f

• 설명 W레지스터의 값과 f레지스터의 값을 AND 연산한 뒤 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)

• 처리 (W) . AND. (f) -->d

• STAUTS Z

• 사이클 1

+ 14BIT OPCODE

MSB LSB

00 0101 dfff ffff

• 사용예 ANDWF 12H,1 (기계어코드: 0592)

실행전	실행후
(W) = 0AAH (12H) = 0FH	(W) = 0AAH (12H) = 0AH

BCF f, b Bit Clear f

• 설명 f레지스터의 비트b를 0으로 만든다.

• 처리 0--> f<b>

• STAUTS 없음(이 말의 의미는 이 명령어는 STATUS 에 영향을 주지 않는다는 의미)

• 사이클 1

+ 14BIT OPCODE

MSB LSB

01 00bb bfff ffff

• 사용 예 BCF 12H,7 (기계어코드: 1392)

실행전	실행후
(12H) = 0FFH	(12H) = 07FH

실행 전 BF1 = B'11111111'(=0FFH)

실행 후 BF1 = B'01111111'(=07FH)

BSF f, b Bit Set f

- 설명 f레지스터의 비트 b를 1로 만든다.
- 처리 1--> f<b>
- STATUS 없음
- 사이클 1
- 14BIT OPCODE

MSB LSB

01 01bb bfff ffff

- 사용 예 BSF 12H,7 (기계어코드: 1792)

실행전	실행후
(12H) = 00H	(12H) = 080H

실행 전 12H = B'00000000'(=00B)

실행 후 12H = B'10000000'(=80H)

BTFSC f,b Bit Test, Skip if Clear

- 설명 f레지스터의 비트 b가 0이면, 이 명령어 다음에 있는 명령어를 건너뛴다.
- 처리 Skip if (f<b>) = 0
- STATUS 없음
- 사이클 1 또는 2(skip 하는 경우 2 cycle)
- 14BIT OPCODE

MSB LSB

01 10bb bfff ffff

- 사용 예 BTFSC 12H, 5 (기계어코드: 1A92)

GOTO LOOP

GOTO EXIT

만약 12H의 데이터 메모리의 비트 5가 1이면 GOTO LOOP 명령을 실행하고, 0이면 GOTO EX IT를 실행한다. 분기가 일어났을 때(비트가 0일 때) 실행 사이클은 2사이클이 되고, 분기가 일어나지 않을 때 (비트가 1일 때) 실행 사이클은 1사이클이 된다.

BTFSS f,b Bit Test, Skip if Set

- 설명 f레지스터의 비트 b가 1이면, 이 명령어 다음에 있는 명령어를 건너뛴다.
- 처리 Skip if(f<b>) = 1

• STATUS 없음
• 사이클 1 또는 2
• 14BIT OPCODE

MSB LSB
01 11bb bfff ffff

• 사용 예

```
BTFSS   12H, 7      (기계어코드: 1F92)
GOTO    LOOP
GOTO    EXIT
```

만약 12H의 비트 7이 1이면 GOTO EXIT명령을 실행하고, 0이면 GOTO LOOP를 실행한다

CALL k Subroutine Call

• 설명 k 주소에 위치하는 서브 루틴을 호출하는 명령이다. 먼저 복귀할 어드레스를 특별한 용도로 사용되는 메모리(이것을 스택이라함)에 저장하고 지정한 번지 k로 분기한다.
• 처리

```
(PC) +1 --> TOS
k --> PC<10:0>
(PCLATCH<4:3>) --> PC<12:11>
```

• operands 0 ≤ k ≤ 2047
• STATUS 없음
• 사이클 2
• 14BIT OPCODE

MSB LSB
10 0kkk kkkk kkkk

• 사용 예

```
LP1     명령어
LP2     CALL    123H        (기계어코드: 2123)
LP3     명령어
```

실행전	실행후
PC = Address LP2	PC = Address XY TOS = Address LP3

* TOS는 top of stack을 의미하며 RETURN 류의 명령어에 의해서 가장 빨리 사용되는 영역에 되돌아올 주소가 저장된다.

☞ 명령어의 k 값의 범위가 11 bit 만을 가지므로 이것보다 더 큰 값을 가지는 경우

는 사용상에 조심을 해야한다. 만약 큰 값을 가지게 하려면 PCLATCH register의 bit4,3이 PC의 bit 12,11과 연관되어 있으므로 이 값을 강제로 변경해 주어야 한다.

CLRF f Clear f

- 설명 F레지스터를 0으로 만들고 Z플래그를 1로 만든다
- 처리 00H --> f
  1 -->Z
- STATUS Z
- 사이클 1
- 14BIT OPCODE

  MSB LSB

  00 0001 1fff ffff

- 사용 예 CLRF 12H (기계어코드: 0192)

실행전	실행후
(12H) = 0AAH	(12H) = 00H

CLRW Clear W Register

- 설명 W레지스터를 0으로 만들고 Z플래그를 1로 만든다.
- 처리 00H --> W
  1 --> Z
- STATUS Z
- 사이클 1
- 14BIT OPCODE

  MSB LSB

  00 0001 0000 0011

- 사용 예 CLRW (기계어코드: 0103)

실행전	실행후
W=0AAH	W=00H

CLRWDT Clear Watchdog Time

• 설명 워치독 타이머의 값을 0으로 만들고, 워치독 프리 스케일러의 값도 0으로 만든다. STATUS레지스터의 TO, PD비트는 1로 만든다.

• 처리 00H --> WDT
0 --> WDT prescaler
1 --> TO
1 --> PD

• STATUS TO, PD

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 0000 0110 0100

• 사용 예 CLRWDT (기계어코드: 0064)

COMF f,d Complement f

• 설명 f레지스터의 내용을 반전시켜 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)

• 처리 .NOT. (f) --> d

• STATUS Z

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 1001 dfff ffff

• 사용예 COMF 12H, 1 (기계어코드: 0992)

실행전	실행후
W = 034H (12H) = 05AH	W = 034H (12H) = 0A5H

실행 전 (12H) = B'10101010' (=05AH)
실행 후 (12H) = B'01010101' (=0A5H)
즉, 1은 0이 되고, 0은 1이 된다.

DECF f,d Decrement f

• 설명 f레지스터의 내용을 하나 감소시켜 그 결과를 d로 보낸다.(d가 "0"이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)

• 처리 (f)-1 --> d

• STATUS Z

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 0011 dfff ffff

• 사용예 DECF 12H, 1 (기계어코드: 0392)

실행전	실행후
(12H) = 11H	(12H) = 10H

DECFSZ f,d Dec, f Skip if Zero

• 설명 f레지스터의 내용을 하나 감소시켜 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.) 그리고 감소시킨 결과가 0이면 다음 명령을 스킵한다.

• 처리 (f)-1 -->d ; Skip if Zero

• STATUS 없음

• 사이클 1 또는 2

• 14BIT OPCODE

MSB LSB

00 1011 dfff ffff

• 사용예

```
DECFSZ 12H, 1          (기계어코드: 0B92)
GOTO   LOOP1
GOTO   LOOP2
```

만약 (12H)을 감소시킨 후의 결과가 0이면 GOTO LOOP2명령을 실행하고, 0이 아니면 GOTO LOOP1를 실행한다.

GOTO k Unconditional Branch

• 설명 번지 k번지로 무조건 분기한다.

• 처리 k --> pc<10:0>

(PALATCH<4:3>)-->PC<12:11>

- operands 0 ≤ k ≤ 2047
- STATUS 없음
- 사이클 2
- 14BIT OPCODE

MSB LSB

10 1kkk kkkk kkkk

- 사용 예 YY1 GOTO 123H (기계어코드: 2923)

실행전	실행후
PC = Address YY1	PC = Address 123H

☞ CALL 명령어와 같이 명령어의 k 값의 범위가 11 bit 만을 가지므로 이것보다 더 큰 값을 가지는 경우는 사용상에 조심을 해야한다. 만약 큰 값을 가지게 하려면 **PCLATCH register 의 bit4,3이 PC의 bit 12,11과 연관되어 있으므로 이 값을 강제로 변경해 주어야 한다.**

INCF f,d Increment f

- 설명 f레지스터의 내용을 하나 증가시켜 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)
- 처리 (f)+1 --> d
- STATUS Z
- 사이클 1
- 14BIT OPCODE

MSB LSB

00 1010 dfff ffff

- 사용예 INCFB 12H, 1 (기계어코드: 0A92)

실행전	실행후
(12H) = 11H	(12H) = 12H

INCFSZ f, d Inc. f Skip if Zero

- 설명 f레지스터의 내용을 하나 증가시켜 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)그리고 증가시킨 결과가

0이면 다음 명령을 수행한다.

- 처리 (f) + 1 --> d ; Skip if Zero
- STATUS 없음
- 사이클 1 또는 2
- 14BIT OPCODE

MSB LSB

00 1111 dfff ffff

- 사용예

```
INCFSZ    12H, F        (기계어코드: 0F92)
GOTO      LOOP1
GOTO      LOOP2
```

만약 (12H)을 증가시킨 후의 결과가 0이면 GOTO LOOP2명령을 실행하고, 0이 아니면 GOTO LOOP1을 실행한다.

IORWL k OR Literal with W

- 설명 W레지스터 값과 상수 k를 OR연산한 뒤 그 결과를 W레지스터에 저장한다.
- 처리 (W) OR k --> W
- operands 0 ≤ k ≤ 255
- STATUS Z
- 사이클 1
- 14BIT OPCODE

MSB LSB

11 1000 kkkk kkkk

- 사용예

```
IORWL     0FH           (기계어코드: 380F)
```

실행전	실행후
W = 0AAH	W = 0AFH

이를 비트로 표시해 보면 다음과 같다.

실행 전 W = B'10101010'(=0AAH)

상수 k = B'00001111'(=00FH)

실행 후 W = B'10101111'(=0AFH)

IORWF f,d OR W with f

- 설명 W레지스터 값과 f레지스터 값을 OR연산한 뒤 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 F레지스터에 저장한다.)
- 처리 (W) . OR. (f) --> d
- STATUS Z
- 사이클 1
- 14BIT OPCODE

MSB LSB

00 0100 dfff ffff

- 사용예 IORWF 12H,1 (기계어코드: 0492)

실행전	실행후
W = 0BBH	W = 0BBH
(12H) =00FH	(12H)=0AFH

MOVLW k MOVE Literal to W

- 설명 상수 k를 W레지스터에 저장한다.
- 처리 k --> W
- operands 0 ≤ k ≤ 255
- STATUS 없음
- 사이클 1
- 14BIT OPCODE

MSB LSB

11 00xx kkkk kkkk (x는 don,t care)

- 사용예 MOVLW 012H (기계어코드: 3012)

실행전	실행후
W = ??H	W = 012H

MOVF f,d MOVE f to d

- 설명 f레지스터 값을 d로 보낸다. (d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.
- 처리 (f) --> d
- STATUS Z

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 1000 dfff ffff

• 사용예 **MOVF 12H, 0** **(기계어코드: 0812)**

실행전	실행후
W = 00H	W = 0FH
(12H) = 0FH	(12H) = 0FH

MOVWF f MOVE W to f

• 설명 W레지스터의 값을 f레지스터에 저장한다.

• 처리 **(W) --> f**

• STATUS **없음**

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 0000 1fff ffff

• 사용예 **MOVWF 12H** **(기계어코드: 0092)**

실행전	실행후
W = 012H	W = 012H
(12H) = ??H	(12H) = 12H

NOP No Operation

• 설명 아무 처리도 수행하지 않고 1사이클만 소비한다.

• 처리 **No Operation**

• STATUS **없음**

• 사이클 1

• 사용예 **NOP** **(기계어코드: 0000)**

• 14BIT OPCODE

MSB LSB

00 0000 0xx0 0000

NOP 명령은 주로 시간 지연의 목적으로 사용한다.

RETFIE Return from Interrupt

- 설명 리턴 명령이다. 가장 최근에 CALL 한 다음 명령어로 되돌아간다. 단순한 RETURN 명령어와의 차이는 GIE(global interrupt enable: INTCON의 bit 7)비트를 1로 만들어 준다는 것이며, 따라서 인터러트 처리 루틴의 마지막에서 만 사용한다.
- 처리 TOS --> PC
  1 --> GIE
- STATUS 없음
- 사이클 2

- 사용예 RETFIE (기계어코드: 0009)

인터럽트 루틴으로부터 메인 루틴으로의 리턴에 사용하는 명령으로 리턴과 동시에 다시 인터럽트를 받아드릴 수 있게 GIE(global interrupt enable: INTCON의 bit 7) 비트를 1로 만들어 준다.

RETWL k Return Literal to W

- 설명 상수 k를 W레지스터에 저장하고 가장 최근에 CALL 한 다음 명령어로 되돌아간다.
- 처리 k --> W
  TOS --> PC
- operands 0 ≤ k ≤ 255
- STATUS 없음
- 사이클 2
- 14BIT OPCODE

  MSB LSB

  11 01xx kkkk kkkk

- 사용예 RETLW 012H (기계어코드: 3412)

실행전	실행후
W=??H	W=12H PC=TOS

이 명령은 부 프로그램에서 조건에 따른 어떤 결과를 가지고 주 프로그램

으로 되돌아 올 때 사용하면 편리하다.

RETURN Return from Subroutine

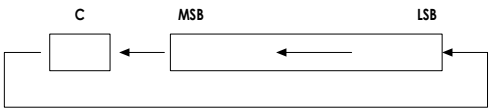
- 설명 단순 리턴 명령이다. 가장 최근에 CALL 한 다음 명령어로 되돌아간다.
- 처리 TOS --> PC
- STATUS 없음
- 사이클 2
- 사용예 RETURN (기계어코드: 0008)

RLF f,d Rotate Left f with Carry

- 설명 C플래그와 함께 f레지스터의 값을 왼쪽으로 회전시키고 그 결과를 d로 보낸다.(d가 '0' 이면 W 레지스터에 저장하고, '1' 이면 F 레지스터에 저장 한다.
- 처리 f<n> --> d<n+1>
  f(7) --> C
  C --> d<0>
- STATUS C
- 사이클 1
- 14BIT OPCODE

  MSB LSB

  00 1101 dfff ffff



RLF의 동작 : C를 포함하여 9 BIT가 됨

- 사용예 RLF 12H,1 (기계어코드: 0D92)

실행전	실행후
(12H) = 8FH C = 0	(12H) = 1EH C = 1

이를 비트로 표시해 보면 다음과 같다.

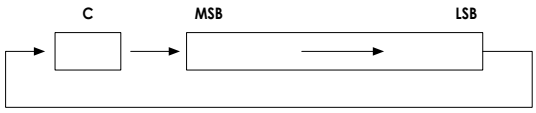
실행 전 (12H) = 10001111B(=8FH), C=0

실행 후 (12H) = 00011110B(=3EH), C=1

RRF f,d Rotate Right f whit Carry

- 설명 C플래그와 함께 f레지스터의 값을 오른쪽으로 회전시키고 그 결과를 d로 보낸다.(d가 '0'이면 W레지스터에 저장하고, '1'이면 F레지스터에 저장한다).
- 처리 f<n> --> d<n-1>

f(0) --> C
C --> d<7>



RRF의 동작 : C를 포함하여 9 BIT가 됨

• STATUS C
• 사이클 1

• 14BIT OPCODE

MSB LSB
00 1100 dfff ffff

• 사용예 **RRF 12H,0** **(기계어코드: 0C12)**

실행전	실행후
(12H) = 8FH W=??H C = 0	(12H) = 8FH W=47H C = 1

이를 비트로 표시해 보면 다음과 같다.
실행 전 (12H) = B'10001111'(=8FH), C=0
실행 후 (12H) = B'01000111'(=47H), C=1

SLEEP Go Sleep Modes

• 설명 Power down mode로 들어가게 만드는 명령어이다.
• 처리 00H --> WDT
00H --> WDT prescaler
1 --> TO
0 --> PD
• STATUS TO, PD
• 사이클 1

• 사용예 **SLEEP** **(기계어코드: 0063)**

프로세서는 슬립모드로 진입하고, 오실레이터 발진은 중단된다. 따라서 마이크로 프로세서가 동작을 멈추므로 전력 소모가 아주 적어져 배터리의 수명을 증가시킬 수 있다.

☞ SLEEP MODE에서 벗어나는 길은:

1) External reset input on /MCLR pin
2) WDT wake-up(if WDT was enabled)
3) Interrupt from RB)/INT pin, RB port change, or data EEPROM write complete)

SUBLW k Sub Literal from W

- 설명 상수 k에서 W레지스터의 값을 뺀 후 그 결과를 W레지스터에 저장한다.
- 처리 k - (W) --> W
- operands 0 ≤ k ≤ 255
- STATUS C, DC, Z
- 사이클 1
- 14BIT OPCODE

MSB LSB
11 110x kkkk kkkk

- 사용예 SUBLW 02H (기계어코드: 3C02)

실행전	실행후
W = 01H C=?, DC=?, Z=?	W = 01H C=1, DC=1, Z=0

실행전	실행후
W = 02H C=?, DC=?, Z=?	W = 00H C=1, DC=1, Z=1

실행전	실행후
W = 03H C=?, DC=?, Z=?	W = 0FFH C=0, DC=0, Z=0

☞ STATUS에서 보면 C, DC는 연산과정에서 자리빌림(borrow)이 발생하지 않으면 '1'로 되고 자리빌림이 일어나면 '0'이 된다.
☞ 자리빌림을 자리올림으로 해석해 보면, - 값을 2의 보수 연산으로 변경하여 더하기로 하

면 되고, 그 결과 자리올림이 생기면 C, DC를 '1'로 설정하면 된다. 즉 C, DC가 '1' 이라는 의미는 다시 뺄셈으로 보면 자리빌림이 발생하지 않았다는 것이다.

SUBWF f,d Sub W to f

• 설명 f레지스터의 값에서 W레지스터의 값을 뺀 뒤 그 결과를 d로 보낸다. (d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)

• 처리 (f) - (W) --> d

• STATUS C, DC, Z

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 0010 dfff ffff

• 사용예 SUBWF 12H,1 (기계어코드: 0292)

실행전	실행후
W = 01H, C =?, DC=?, Z=? (12H) = 00H	W = 01H, C = 0, DC = 0, Z=0 (12H) = 0FFH

자리 빌림이 발생하였으므로 캐리 플래그는 '0'이 된다.

SUBWF BF1,0

실행전	실행후
W = 22H, C =?, DC=?, Z=? (12H) = 22H	W = 00H, C = 0, DC = 0, Z=1 (12H) = 22H

자리 빌림이 발생하지 않았으므로 캐리 플래그는 '1'이 된다.

SWAPF f,d Swap f

• 설명 f레지스터의 상위 니블(4BIT)과 하위 니블(4BIT)을 서로 맞바꾼 후 그 결과를 d로 보낸다. (d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)

• 처리 f<0:3> --> d<4:7>

f<4:7> --> d<0:3>

• STATUS 없음

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 1110 dfff ffff

• 사용예 SWAPF 12H, 1 (기계어코드: 0E92)

실행전	실행후
(12H) = 05AH	(12H) = 0A5H

XORLW k Ex.OR Literal with W

• 설명 W레지스터의 값과 상수 k를 XOR연산한 뒤 그 결과를 W레지스터에 저장한다.

• 처리 (W) .XOR. K --> W

• operands 0 ≤ k ≤ 255

• STATUS Z

• 사이클 1

• 14BIT OPCODE

MSB LSB

11 1010 kkkk kkkk

• 사용예 XORLW 0FH (기계어코드: 2A0F)

실행전	실행후
W = 0AAH	W = 0A5H

이를 비트로 표시해 보면 다음과 같다

실행 전 W = B'10101010'(=0AAH)

상수 k = B'00001111'(=0FH)

실행 후 W = B'10100101'(=0A5H)

XORWF f,d Ex.OR with f

• 설명 W레지스터의 값과 f레지스터의 값을 XOR 연산한 뒤 그 결과를 d로 보낸다. (d가 '0'이면 W레지스터에 저장하고, '1'이면 f레지스터에 저장한다.)

• 처리 (W) .XOR. (f) --> d

• STATUS Z

• 사이클 1

• 14BIT OPCODE

MSB LSB

00 0110 dfff ffff

• 사용예 **XORWF** 12H,1 **(기계어코드: 0692)**

실행전	실행후
W = 0AAH (12H) = 0FH	W = 0AAH (12H) = 0A5H