Mikroişlemciler

Ders -2

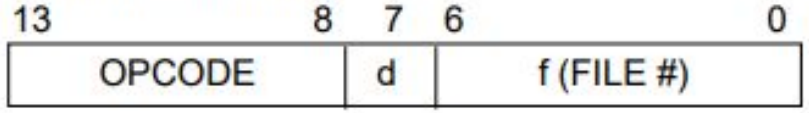
Instruction Set

- RISC mimari
- 35 komut
- 35 komut tanımı için komut pattern ‘de 6 bit ‘e ihtiyaç vardır (her zaman 6 bit olmak zorunda değil)
- Komut uzunluğu (aynı zamanda Word uzunluğu) 14 bittir
- ALU ve W register 8 bit işlem yapar

TABLE 13-2: PIC16F87X INSTRUCTION SET

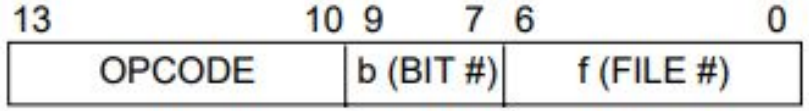
Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected
		BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff	
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff	
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z
MOVWF	f	Move W to f	1	00	0000	lfff	ffff	
NOP	-	No Operation	1	00	0000	0xx0	0000	
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff	
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z
		BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff	
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff	
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff	
		LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk	
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk	
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk	
RETFIE	-	Return from interrupt	2	00	0000	0000	1001	
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk	
RETURN	-	Return from Subroutine	2	00	0000	0000	1000	
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z

Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

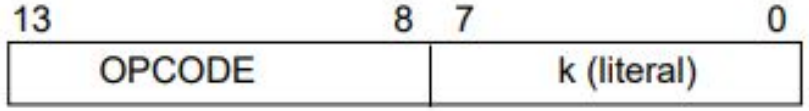
Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

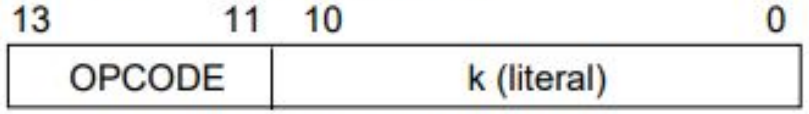
Literal and control operations

General



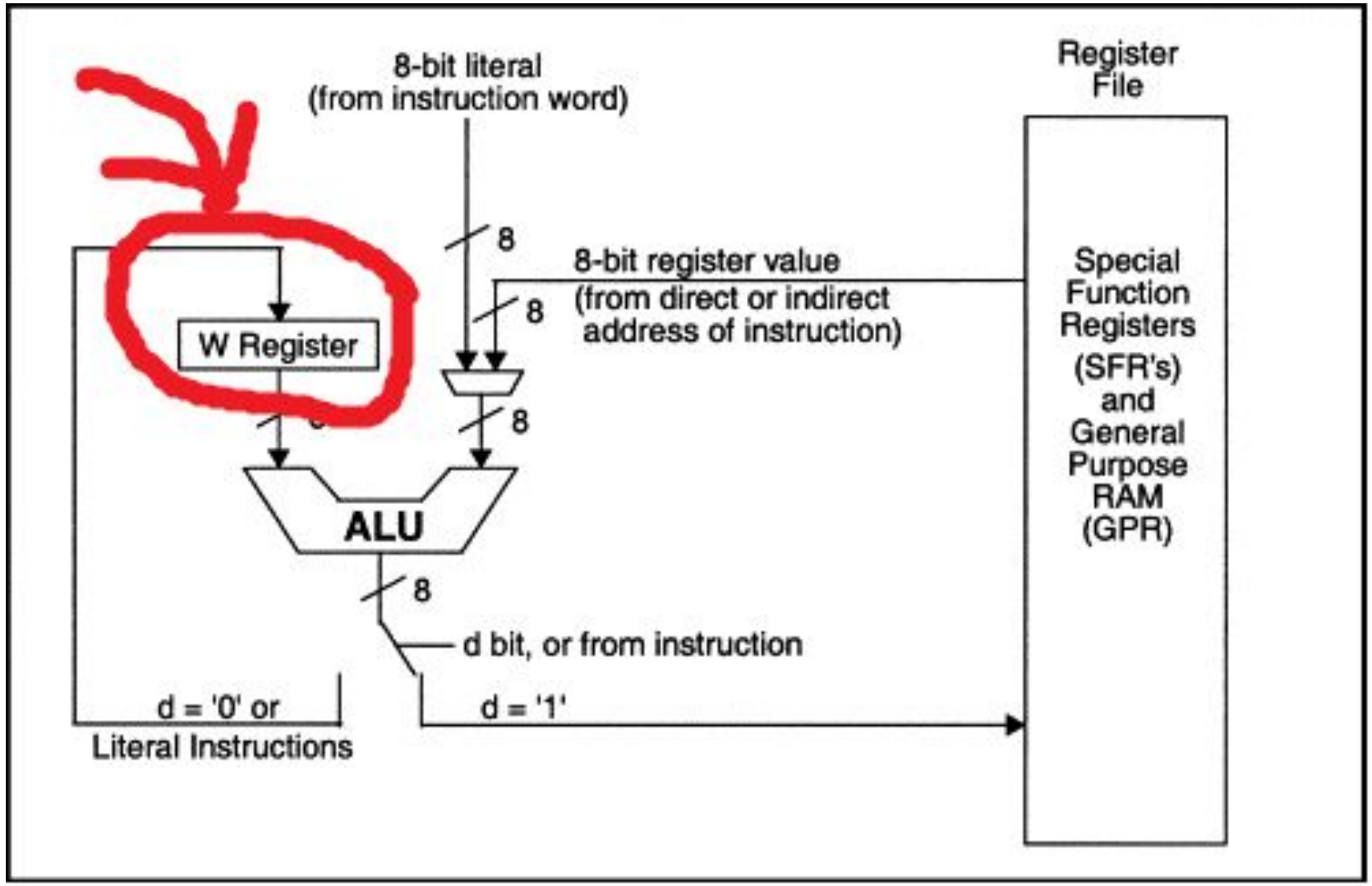
k = 8-bit literal (immediate) value

`CALL` and `GOTO` instructions only



k = 11-bit literal (immediate) value

Working Register (W) ‘nı Anlayalım



ADDLW

ADDLW — Add Literal and W

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	$(W) + k \rightarrow W$
Status Affected:	C, DC, Z
Encoding:	11 | 111x | kkkk | kkkk
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.
Words:	1
Cycles:	1

Example1 `ADDLW 0x15`

Before Instruction

W = 0x10

After Instruction

W = 0x25

Example 2 `ADDLW MYREG`

Before Instruction

W = 0x10

Address of `MYREG` † = 0x37

† `MYREG` is a symbol for a data memory location

After Instruction

W = 0x47

Example 4 `ADDLW MYREG`

Before Instruction

W = 0x10

Address of `PCL` † = 0x02

† `PCL` is the symbol for the Program Counter low byte location

After Instruction

W = 0x12

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → destination
Status Affected:	C, DC, Z

Encoding:

00	0111	dfff	ffff

Description: Add the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

- ADDWF RAM_Register,0 ▫ W =W + RAM_Register değeri
- ADDWF RAM_Register,1 ▫ RAM_Register değeri =W + RAM_Register değeri

ANDLW — And Literal with W

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W).AND. (k) → W
Status Affected:	Z

Encoding:

11	1001	kkkk	kkkk

Description: The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1

```
ANDLW   0x5F
```

Before Instruction
W = 0xA3
After Instruction
W = 0x03

```
; 0101 1111   (0x5F)
; 1010 0011   (0xA3)
;----------   ------
; 0000 0011   (0x03)
```

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W).AND. (f) → destination
Status Affected:	Z
Encoding:	00 | 0101 | dfff | ffff
Description:	AND the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
ANDWF   FSR, 1
Before Instruction                  ; 0001 0111   (0x17)
        W   = 0x17                  ; 1100 0010   (0xC2)
        FSR = 0xC2                  ;-----------  ------
After Instruction                   ; 0000 0010   (0x02)
        W   = 0x17
        FSR = 0x02
```

Example 2

```
ANDWF   FSR, 0
Before Instruction                  ; 0001 0111   (0x17)
        W   = 0x17                  ; 1100 0010   (0xC2)
        FSR = 0xC2                  ;-----------  ------
After Instruction                   ; 0000 0010   (0x02)
        W   = 0x02
        FSR = 0xC2
```

BCF — Bit Clear f

Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → f<b>
Status Affected:	None

Encoding:

01	00bb	bfff	ffff

Description: Bit 'b' in register 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example 1

```
BCF     FLAG_REG, 7
```

Before Instruction

FLAG_REG = 0xC7 ; 1100 0111

After Instruction

FLAG_REG = 0x47 ; 0100 0111

Example 2

```
BCF     INDF, 3
```

Before Instruction

W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x2F

After Instruction

W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x27

BSF — Bit Set f

Syntax: [*label*] BSF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $1 \rightarrow f<b>$

Status Affected: None

Encoding:

01	01bb	bfff	ffff

Description: Bit 'b' in register 'f' is set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example 1

```
BSF     FLAG_REG, 7
```

Before Instruction
FLAG_REG =0x0A ; 0000 1010

After Instruction
FLAG_REG =0x8A ; 1000 1010

Example 2

```
BSF     INDF, 3
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x20

After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x28

BTFSC — Bit Test, Skip if Clear

Syntax: [*label*] BTFSC f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0' then the next instruction is skipped.
If bit 'b' is '0' then the next instruction (fetched during the current instruction execution) is discarded, and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No operation

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example 1

```
HERE    BTFSC   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = address HERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = address TRUE

Case 2: Before Instruction
PC = address HERE
FLAG= xxx1 xxxx
After Instruction
Since FLAG<4>=1,
PC = address FALSE

BTFSS

Bit Test f, Skip if Set

Syntax: [*label*] BTFSS f,b

Operands: $0 \le f \le 127$
$0 \le b < 7$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

01	llbb	bfff	ffff

Description: If bit 'b' in register 'f' is '1' then the next instruction is skipped.
If bit 'b' is '1', then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No operation

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example 1

```
HERE    BTFSS   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = address HERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = address FALSE

CALL — Call Subroutine

Syntax:	[*label*] CALL k
Operands:	$0 \le k \le 2047$
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None

Encoding:

10	0kkk	kkkk	kkkk

Description: Call Subroutine. First, the 13-bit return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH<4:3>. `CALL` is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example 1

```
HERE    CALL   THERE
```

Before Instruction
PC = Address`HERE`
After Instruction
TOS = Address `HERE+1`
PC = Address `THERE`

CLRF — Clear f

Syntax: [*label*] CLRF f

Operands: $0 \le f \le 127$

Operation: $00h \rightarrow f$
$1 \rightarrow Z$

Status Affected: Z

Encoding:

00	0001	1fff	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example 1

```
CLRF    FLAG_REG
```

Before Instruction
FLAG_REG=0x5A
After Instruction
FLAG_REG=0x00
Z = 1

Example 2

```
CLRF    INDF
```

Before Instruction
FSR = 0xC2
Contents of Address (FSR)=0xAA
After Instruction
FSR = 0xC2
Contents of Address (FSR)=0x00
Z = 1

CLRW — Clear W

Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → W 1 → Z
Status Affected:	Z
Encoding:	00 | 0001 | 0xxx | xxxx
Description:	W register is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'W'

Example 1

```
CLRW
```

Before Instruction

W = 0x5A

After Instruction

W = 0x00

Z = 1

COMF

Complement f

Syntax: [*label*] COMF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ destination

Status Affected: Z

Encoding:

00	1001	dfff	ffff

Description: The contents of register 'f' are 1's complemented. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
COMF    REG1, 0
```

Before Instruction
REG1= 0x13
After Instruction
REG1= 0x13
W = 0xEC

DECF — Decrement f

Syntax:	[*label*] DECF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → destination
Status Affected:	Z
Encoding:	00 | 0011 | dfff | ffff
Description:	Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
DECF    CNT, 1
```

Before Instruction
CNT = 0x01
Z = 0
After Instruction
CNT = 0x00
Z = 1

DECFSZ — Decrement f, Skip if 0

Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → destination; skip if result = 0
Status Affected:	None

Encoding:

00	1011	dfff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE      DECFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Case 1:
Before Instruction
PC = address HERE
CNT = 0x01
After Instruction
CNT = 0x00
PC = address CONTINUE

Case 2:
Before Instruction
PC = address HERE
CNT = 0x02
After Instruction
CNT = 0x01
PC = address HERE + 1

GOTO — Unconditional Branch

Syntax: [*label*] GOTO k

Operands: $0 \le k \le 2047$

Operation: $k \rightarrow PC<10:0>$
$PCLATH<4:3> \rightarrow PC<12:11>$

Status Affected: None

Encoding:

10	1kkk	kkkk	kkkk

Description: GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
GOTO THERE
```

After Instruction
PC =Address THERE

INCF — Increment f

Syntax:	[*label*] INCF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → destination
Status Affected:	Z
Encoding:	00 | 1010 | dfff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
INCF    CNT, 1
```

Before Instruction
CNT = 0xFF
Z = 0
After Instruction
CNT = 0x00
Z = 1

INCFSZ — Increment f, Skip if 0

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → destination, skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE      INCFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Case 1:
Before Instruction
PC = address HERE
CNT = 0xFF
After Instruction
CNT = 0x00
PC = address CONTINUE

Case 2:
Before Instruction
PC = address HERE
CNT = 0x00
After Instruction
CNT = 0x01
PC = address HERE + 1

MOVLW — Move Literal to W

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

11	00xx	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1

```
MOVLW   0x5A
```

After Instruction

W = 0x5A

Example 2

```
MOVLW   MYREG
```

Before Instruction

W = 0x10

Address of MYREG † = 0x37

† MYREG is a symbol for a data memory location

After Instruction

W = 0x37

Example 3

```
MOVLW   HIGH (LU_TABLE)
```

Before Instruction

W = 0x10

Address of LU_TABLE † = 0x9375

† LU_TABLE is a label for an address in program memory

After Instruction

W = 0x93

MOVF — Move f

Syntax: [*label*] MOVF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) → destination

Status Affected: Z

Encoding:

00	1000	dfff	ffff

Description: The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If 'd' = 0, destination is W register. If 'd' = 1, the destination is file register 'f' itself. 'd' = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
MOVF    FSR, 0
```

Before Instruction
W = 0x00
FSR = 0xC2
After Instruction
W = 0xC2
Z = 0

Example 2

```
MOVF    FSR, 1
```

Case 1: Before Instruction
FSR = 0x43
After Instruction
FSR = 0x43
Z = 0

Case 2: Before Instruction
FSR = 0x00
After Instruction
FSR = 0x00
Z = 1

MOVWF — Move W to f

Syntax: [*label*] MOVWF f

Operands: $0 \le f \le 127$

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

00	0000	1fff	ffff

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example 1

```
MOVWF   OPTION_REG
```

Before Instruction

OPTION_REG=0xFF
W = 0x4F

After Instruction

OPTION_REG=0x4F
W = 0x4F

NOP — No Operation

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None

Encoding:

00	0000	0xx0	0000

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example :

```
HERE    NOP
```

Before Instruction
PC = address HERE
After Instruction
PC = address HERE + 1

RETFIE — Return from Interrupt

Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

Encoding:

00	0000	0000	1001

Description: Return from Interrupt. The 13-bit address at the Top of Stack (TOS) is loaded in the PC. The Global Interrupt Enable bit, GIE (INTCON<7>), is automatically set, enabling Interrupts. This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	No operation	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
RETFIE
```

After Instruction

PC = TOS

GIE = 1

RETLW — Return with Literal in W

Syntax:	[*label*] RETLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow W$; $TOS \rightarrow PC$
Status Affected:	None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded 13-bit address at the Top of Stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

RETURN — Return from Subroutine

Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Encoding:	00 | 0000 | 0000 | 1000
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	No operation	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE    RETURN
```

After Instruction

PC = TOS

RLF

Rotate Left f through Carry

Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: See description below

Status Affected: C

Encoding:

00	1101	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
RLF     REG1,0
```

Before Instruction

```
REG1=  1110 0110
C   =  0
```

After Instruction

```
REG1=1110 0110
W   =1100 1100
C   =1
```

RRF — Rotate Right f through Carry

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

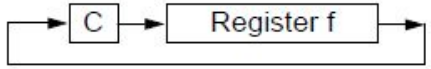
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
RRF          REG1,0
```

Before Instruction

```
REG1= 1110 0110
W   =  xxxx xxxx
C   =  0
```

After Instruction

```
REG1= 1110 0110
```

SUBLW — Subtract W from Literal

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1:

```
SUBLW   0x02
```

Case 1: Before Instruction

```
W = 0x01
C = x
Z = x
```

After Instruction

```
W = 0x01
C = 1       ; result is positive
Z = 0
```

Case 2: Before Instruction

```
W = 0x02
C = x
Z = x
```

After Instruction

```
W = 0x00
C = 1       ; result is zero
Z = 1
```

Case 3: Before Instruction

```
W = 0x03
C = x
Z = x
```

After Instruction

```
W = 0xFF
C = 0       ; result is negative
Z = 0
```

Example 2

```
SUBLW   MYREG
```

Before Instruction

```
W = 0x10
Address of MYREG † = 0x37
```

† MYREG is a symbol for a data memory location

After Instruction

```
W = 0x27
C = 1       ; result is positive
```

SUBWF — Subtract W from f

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → destination

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1: `SUBWF REG1,1`

Case 1: Before Instruction

REG1= 3
W = 2
C = x
Z = x

After Instruction

REG1= 1
W = 2
C = 1 ; result is positive
Z = 0

Case 2: Before Instruction

REG1= 2
W = 2
C = x
Z = x

After Instruction

REG1= 0
W = 2
C = 1 ; result is zero
Z = 1

Case 3: Before Instruction

REG1= 1
W = 2
C = x
Z = x

SWAPF — Swap Nibbles in f

Syntax: [*label*] SWAPF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<3:0>) → destination<7:4>,
(f<7:4>) → destination<3:0>

Status Affected: None

Encoding:

00	1110	dfff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write to destination

Example 1

```
SWAPF   REG, 0
```

Before Instruction

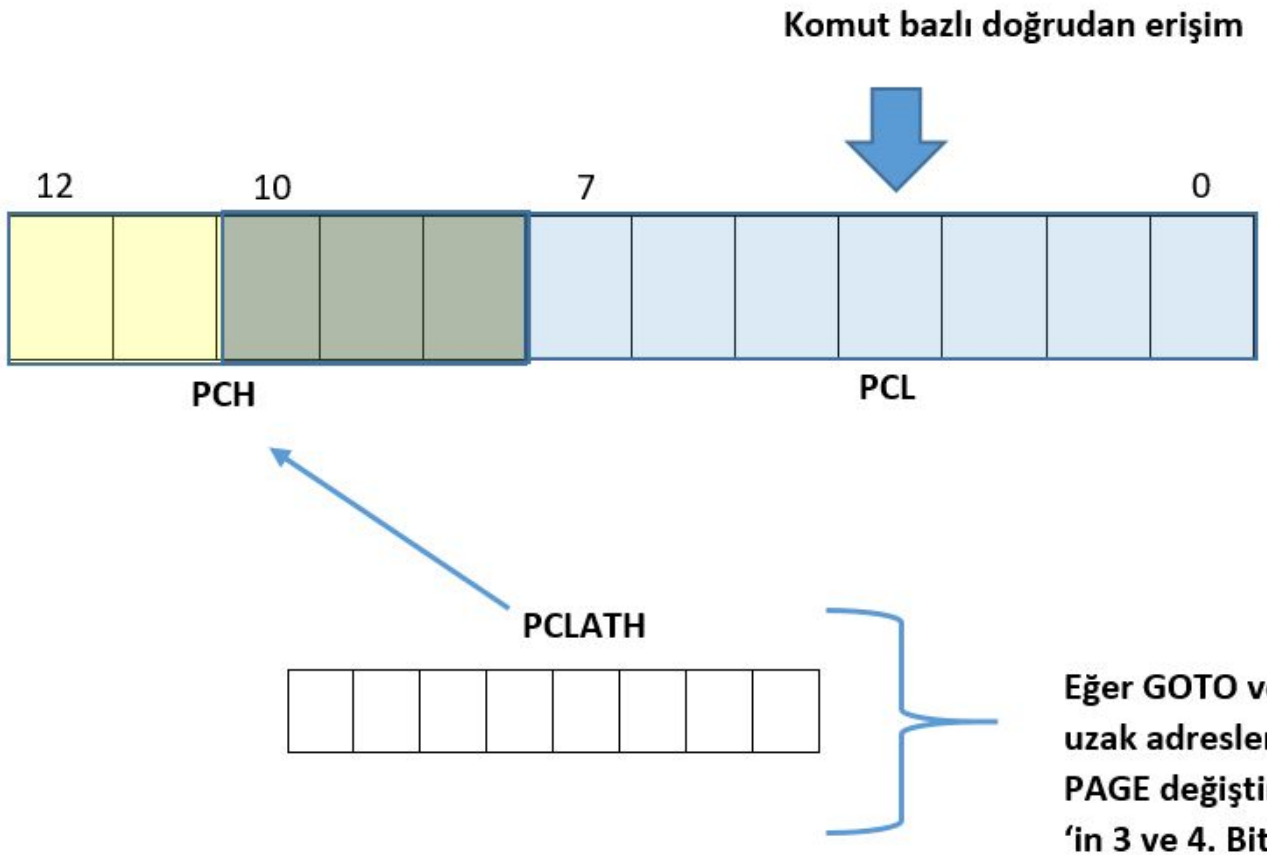
REG1= 0xA5

After Instruction

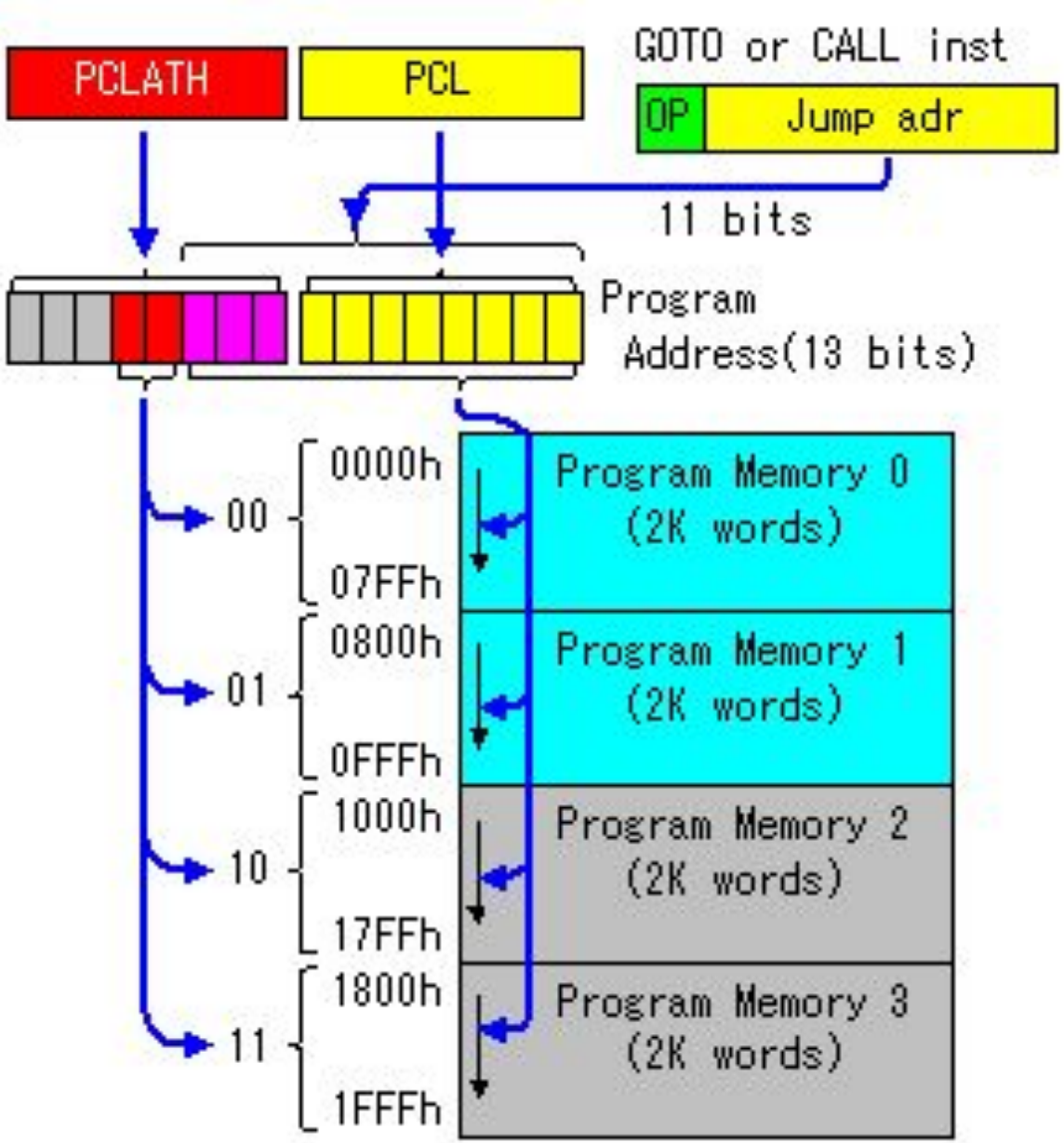
REG1= 0xA5
W = 0x5A

Program Counter

- PC aslında bir RAM register ‘ıdır
- RAM register ‘ları 8 bit peki nasıl PC 13 bit ?

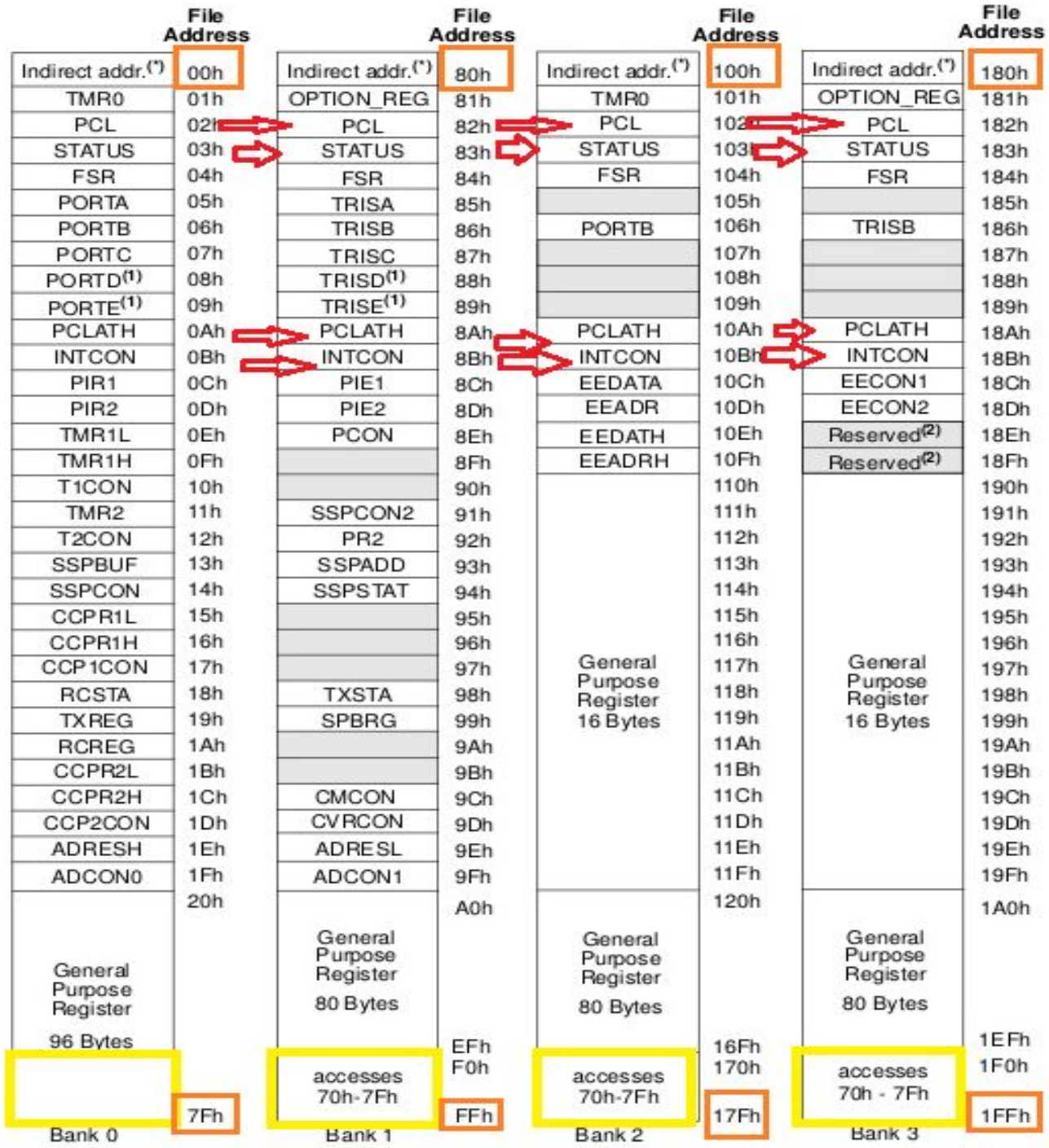


- INTERRUPT hariç !!!



- Her bir PAGE ‘in 2K olduğuna dikkat ediniz (0-2047)

- Dikkat !! Burası FLASH Memory

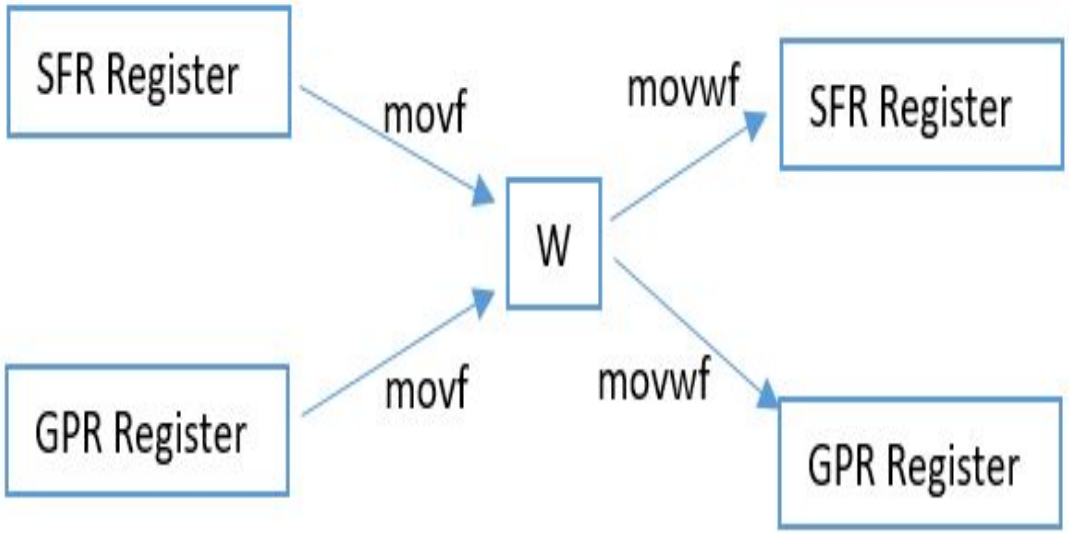


- Burası Register File
- Her biri 128 adres
- 0x00 - 0x7F dikkat edin
- Bazı SFR register ‘lar tüm Bank ‘larda mevcuttur
- Shadowed memory ortak erişim alanı gibi davranır (0x70 ile 0x7F arası ilk PAGE)

IMMIDIATE ADDRESSING

Örnek movlw 0x21

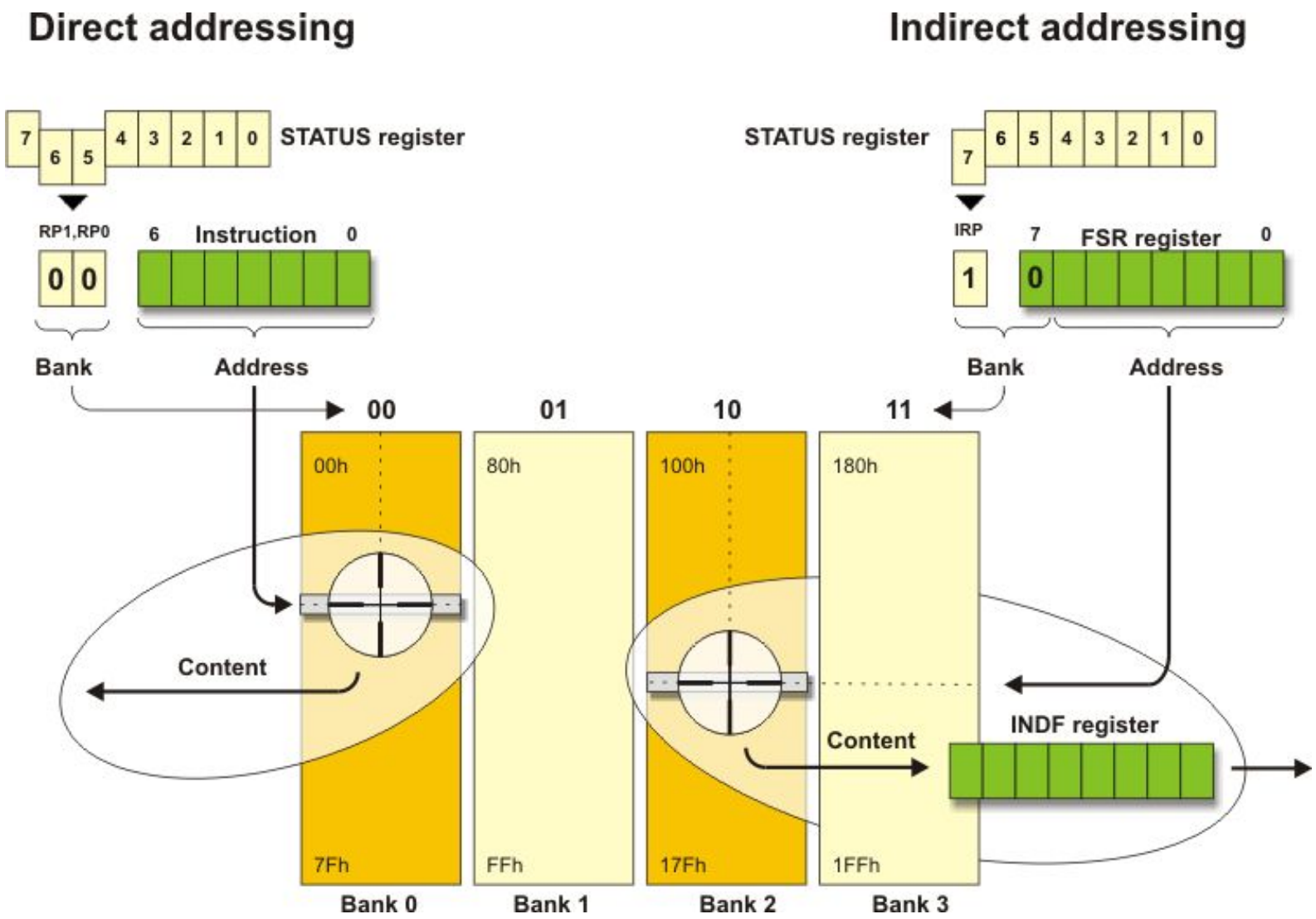
DIRECT ADDRESSING



Örnek movf BizimRegister,0

Örnek movwf BizimRegister2

INDRECT ADDRESING



- FSR Adresi
- INDF ise FSR ‘nin gösterdiği adresteki değeri tutar
- STATUS ‘un IRP si ve FSR ‘nin MSB si bank seçiminde kullanılır
- Array işlemlerinde faydalıdır

Example: Clear all RAM locations from 20h to 7Fh

W Register:

20

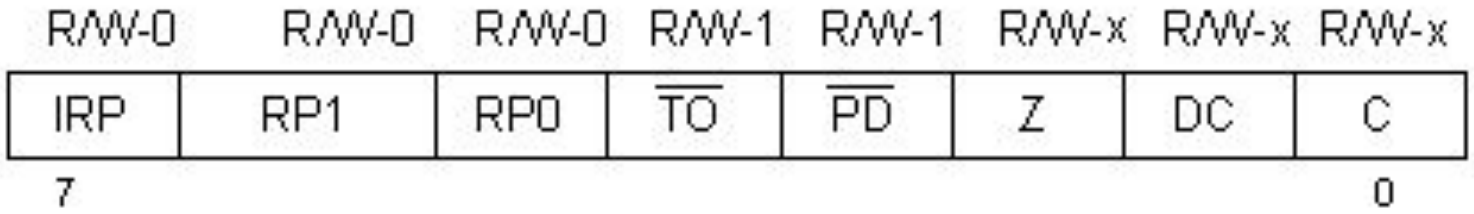
9-Bit Effective Address:

0	0	0	0	0	0	0	0	0
IRP	FSR							

```
        bcf     STATUS,IRP
        movlw   0x20
        movwf   FSR
LOOP    clrf    INDF
        incf    FSR,f
        btfss   FSR,7
        goto    LOOP
        <next instruction>
```

Register File	Address
00	00h : INDF
FF	01h : TMR0
FF	02h : PCL
18	03h : STATUS
80	04h : FSR
00	20h
00	21h
00	22h
00	23h
00	7Dh
00	7Eh
00	7Fh
FF	80h

STATUS Register



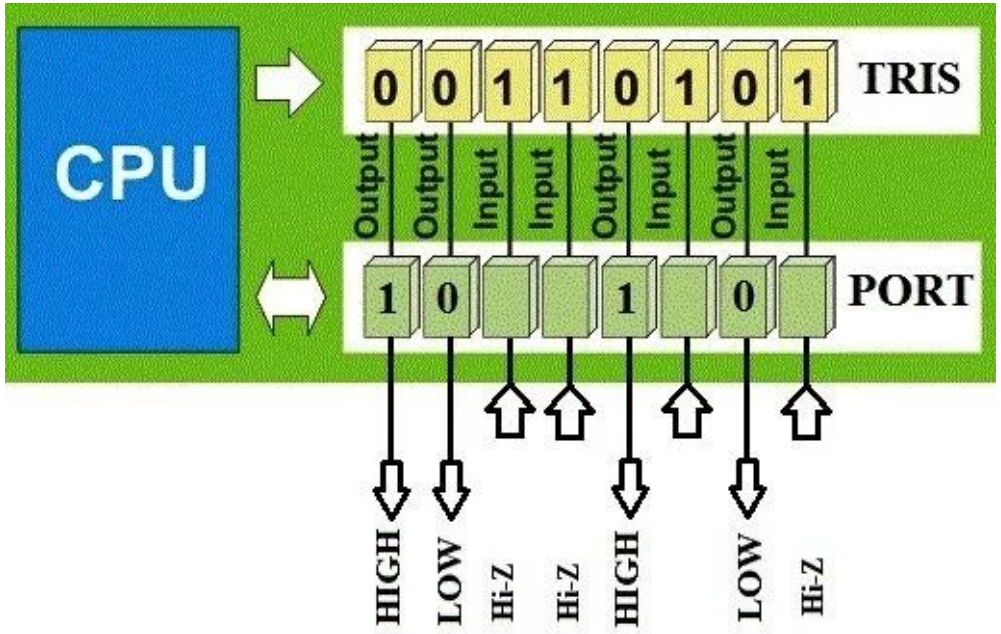
R = Readable bit **W** = Writable bit

U = Unimplemented bit, read as '0' - n = Value at power-on reset

- En önemli register ‘lardan biridir
- PIC ‘in anlık durumu, Aritmetik İşlem Sonuç Durumu ve Bank değiştirme
  - C = ALU sonucu 255 ‘aşarsa 1, çıkarma işleminde sonuç negatif çıkarsa 0 olur (Neden ? Cevaplayınız)
  - İlgili register ‘ın ilk dört bitinin 15 ‘ aşması durumunda 1 olur
  - Z = Herhangi bir komut operasyonunda sonuç 0 olursa (Bazen tetikleme kontrol için de kullanılır)
  - RP0 ve RP1 bitleri Register File ‘da bank değiştiemek için
  - Indrect adreslemede IRP, FSR ‘nin MSB si ile birlikte Bank seçmek için
  - TO ve PD bitleri PIC ‘in ilk kez mi çalıştığı, uykudan mı uyandığı kontrolü için

	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	R/W (x)	Features
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	Features
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	



Portlar

- Tüm portlarda 20mA besleme, 25mA sink akım desteği
- PORTA
  - TRISA ile ayarlanır
  - Buffer desteği
  - A/D desteği
- PORTB
  - TRISB ile ayarlanır
  - Weak-Pull Up desteği
  - TTL tipi
  - Harici kesme (RB0 (DK,YK) ve RB4-RB7) desteği

Portlar

- PortC
  - TRISC ile yönetilir
  - Schimit Trigger tipi sinyal tanımlama
  - Seri haberleşme desteği
- PORTD
  - PORTE ile birlikte paralel haberleşme desteği
- PORTE
  - I/O ve paralel haberleşme desteği