Try to Design and Implementation of a PIC16 compatible microcontroller

Skill up course

PIC16 Microcontroller

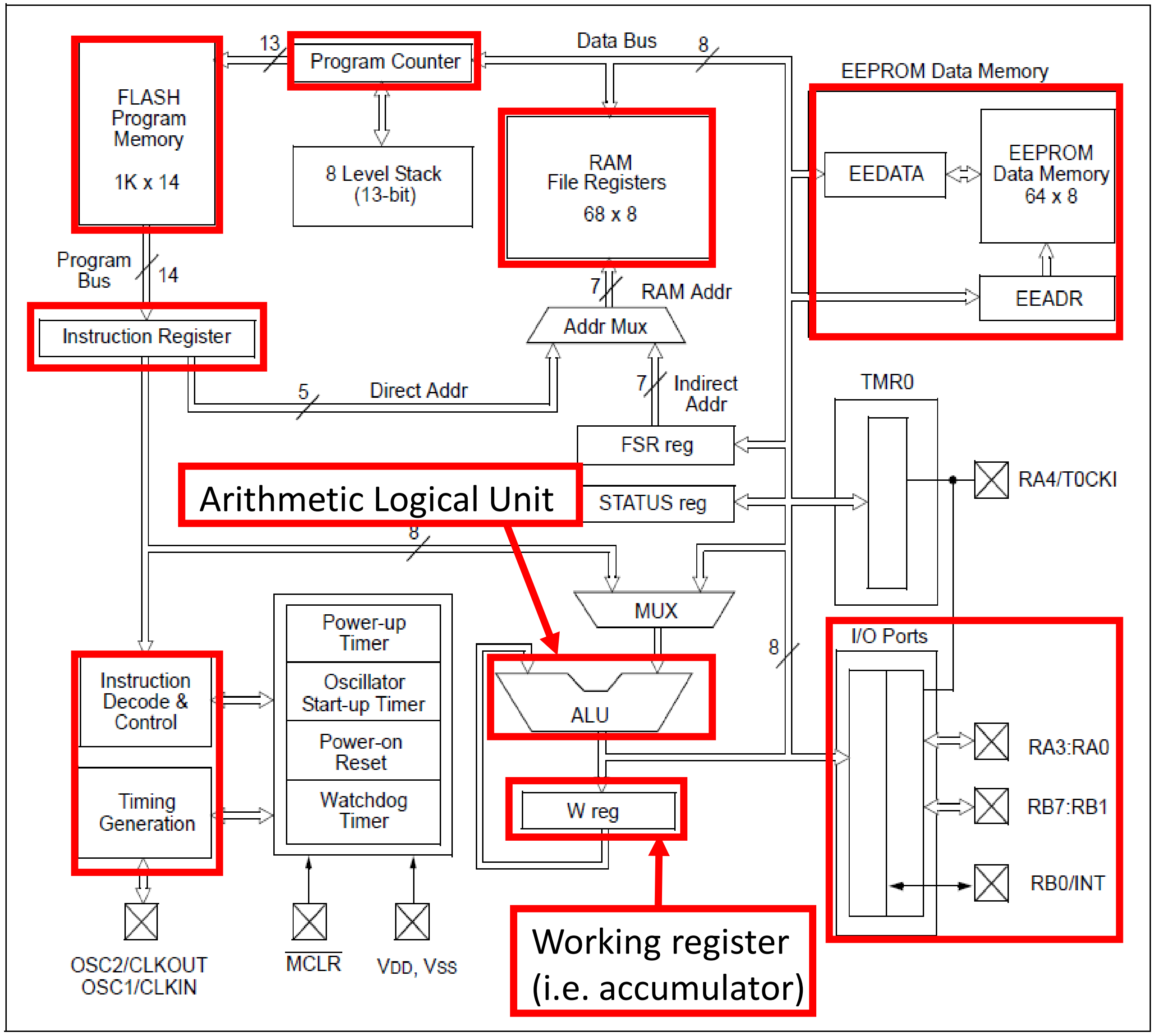
PIC16F84A

by Microchip

Widely used for electronics workshop.

Microcontroller:
It is one chip computer includes of microprocessor, memory and some peripherals.
You can make several controllers.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



Sample program：Store a value into data memory

A constant value is stored into data memory combined with "movlw" and "movwf" instructions

```
movlw  D'10'   Store a constant  10(10) in instruction field to W register
movwf  H'20'   Store a value of W register into address 20(16) of data memory
```

$\underbrace{\underbrace{\text{movlw movwf}}_{\text{mnemonic}}\ \underbrace{\text{D'10' H'20'}}_{\text{operand}}}_{\text{Assembly Language}}$

mnemonic   operand

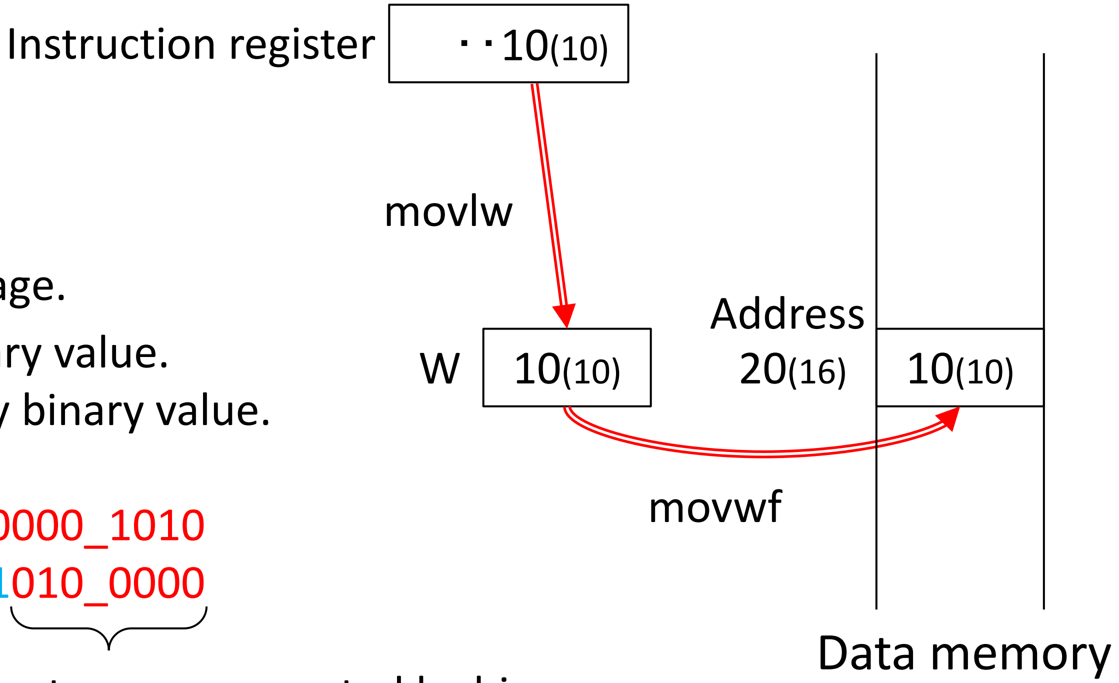
Assembly Language:
It is a language for ease of understanding machine language.

Processor only recognizes binary value.
Program is also represented by binary value.
(Machine Language)

movlw D'10' ⇨ 11_0000_0000_1010
movwf H'20'   00_0000_1010_0000

Machine language is generated by Assembler automatically.

Constants are represented by binary.
$10_{(10)} \rightarrow 1010_{(2)}$, $20_{(16)} \rightarrow 10000_{(2)}$



Behavior of "movlw" instruction

Instruction behavior that a constant in instruction is stored to W register.

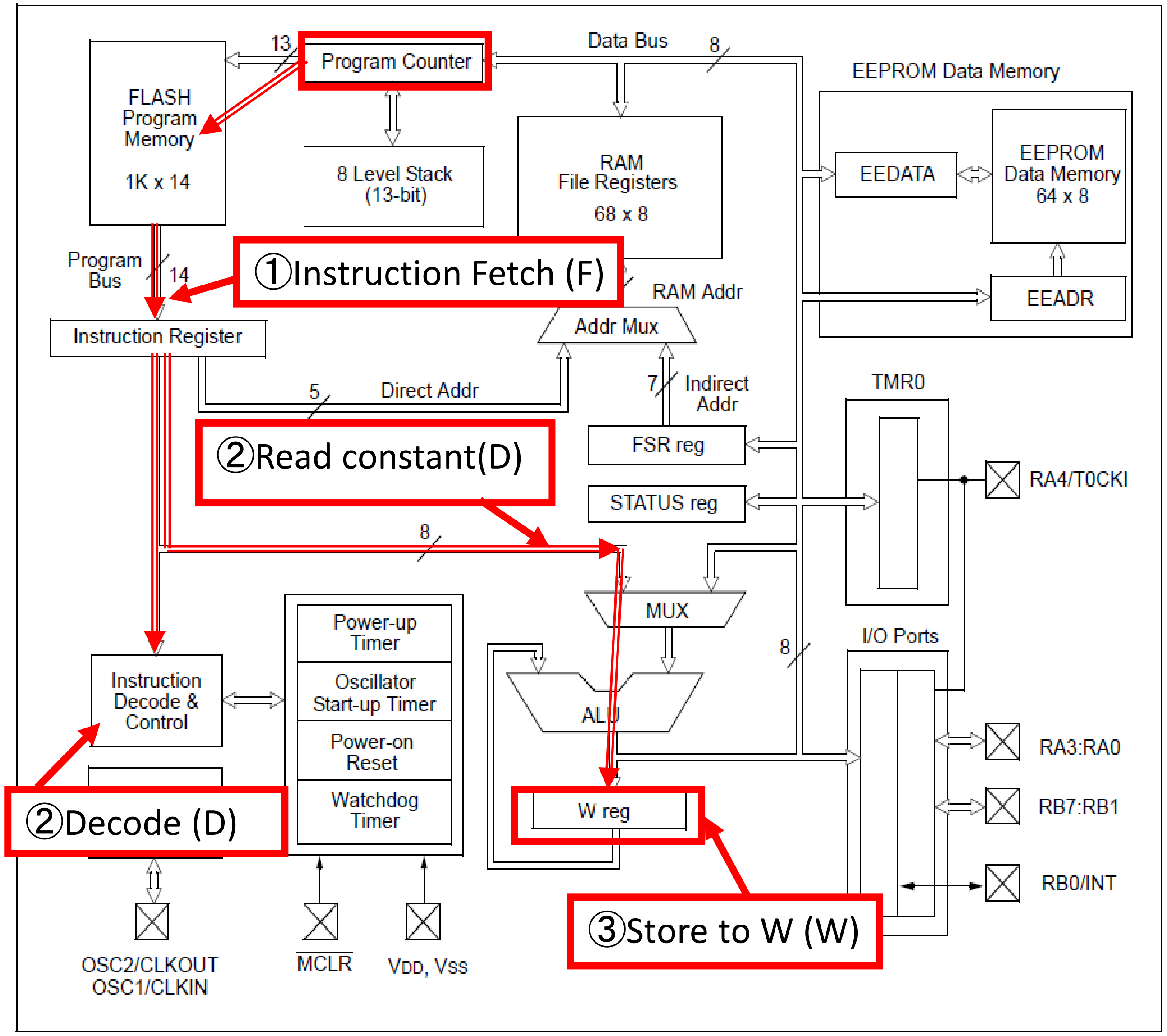
movlw constant

①Instruction pointed out by program counter is stored to instruction register.

②Instruction is decoded. And constant in instruction is ready to be read.

③Store the constant to W register.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



Behavior of "movwf" instruction

Instruction behavior that a value in W register is stored into data memory.

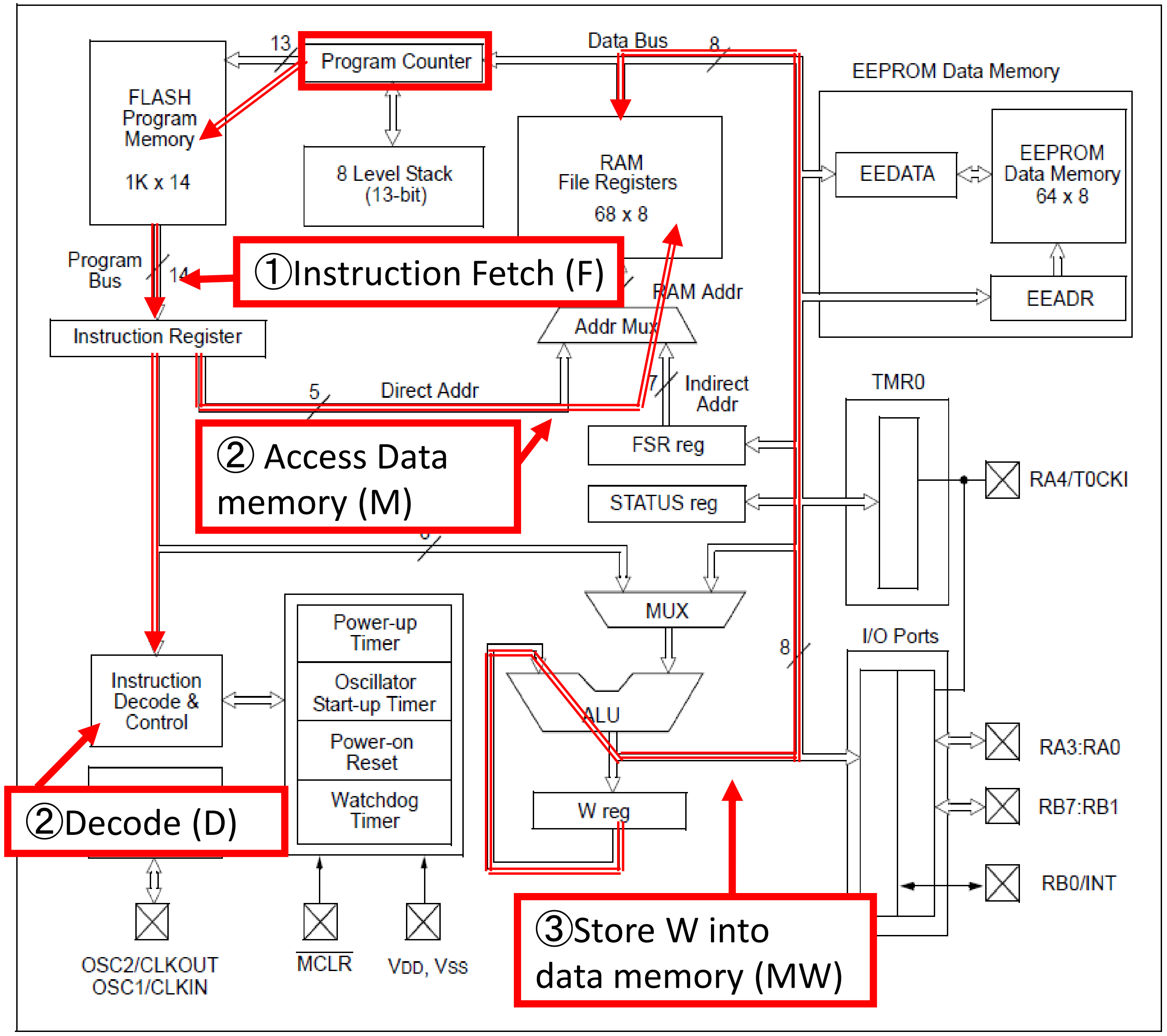
movwf Address

①Instruction pointed out by program counter is stored to instruction register.

② Instruction is decoded. And address for accessing data memory is ready.

③W register value is stored into data memory through the ALU

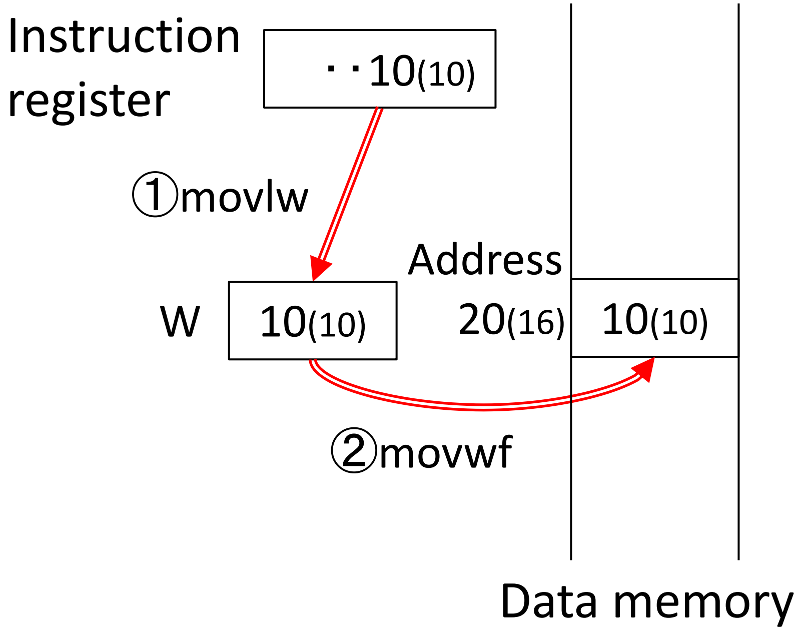
FIGURE 1-1: PIC16F84A BLOCK DIAGRAM

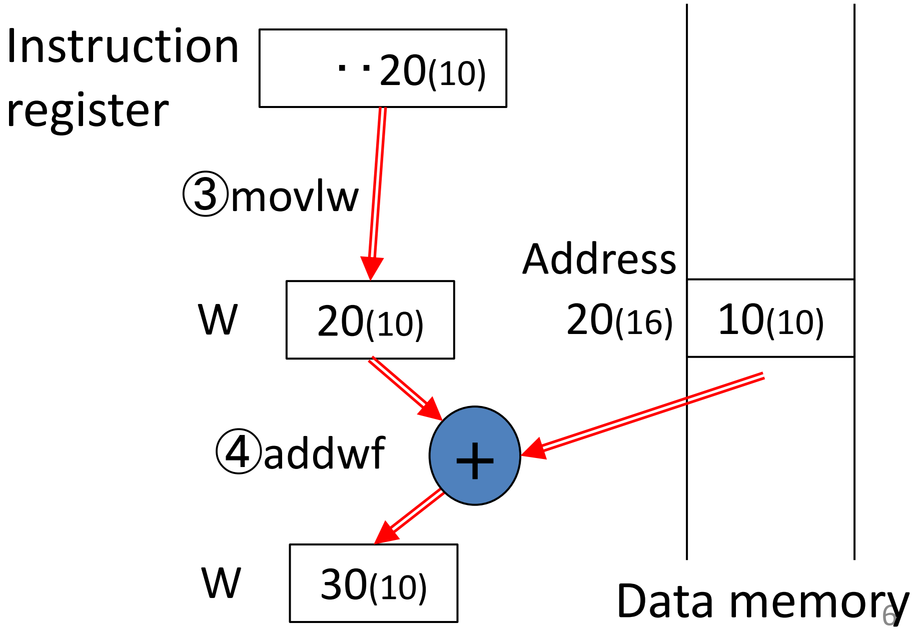


Sample Program：Addition

Add a value in data memory and W register by "addwf" instruction （Example: 10+20=30）

①	movlw D'10'	Store a constant 10(10) in instruction field to W register
②	movwf H'20'	Store a value of W register into address 20(16) of data memory
③	movlw D'20'	Store a constant 20(10) in instruction field to W register
④	addwf H'20',w	Add W register and a value of data memory addressed by 20(20), and the result 20(10) is stored to W register





加算命令の動作例(addwf Address,w)

Instruction behavior that a value in data memory and W register value are added and the result is stored to W register.

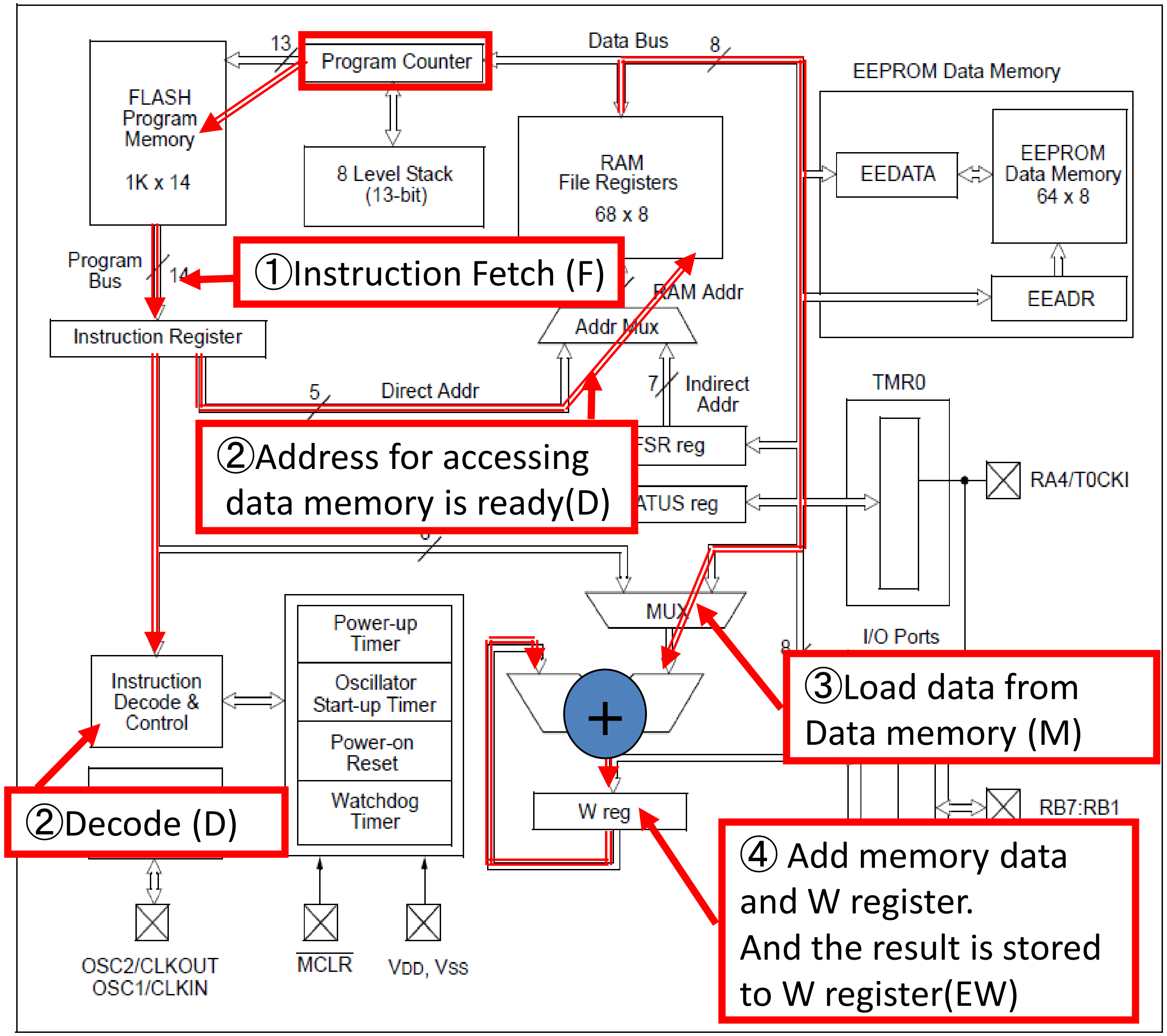
addwf Address, w

①Instruction pointed out by program counter is stored to instruction register.

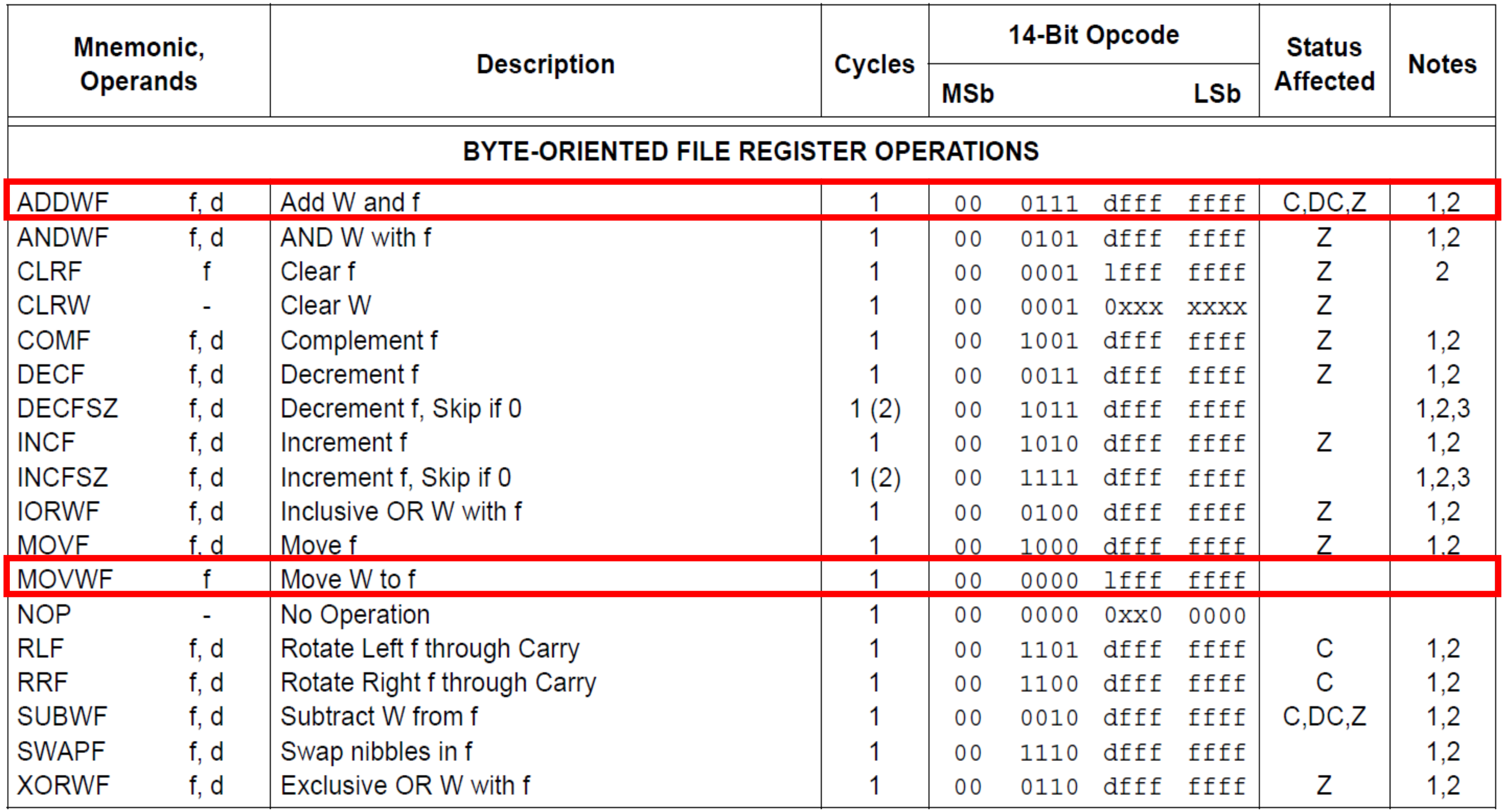
②Instruction is decoded. And address for accessing data memory is ready.

③④Add memory data and W register. And the result is stored to W register.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



Instruction Set of PIC16F84A (1)

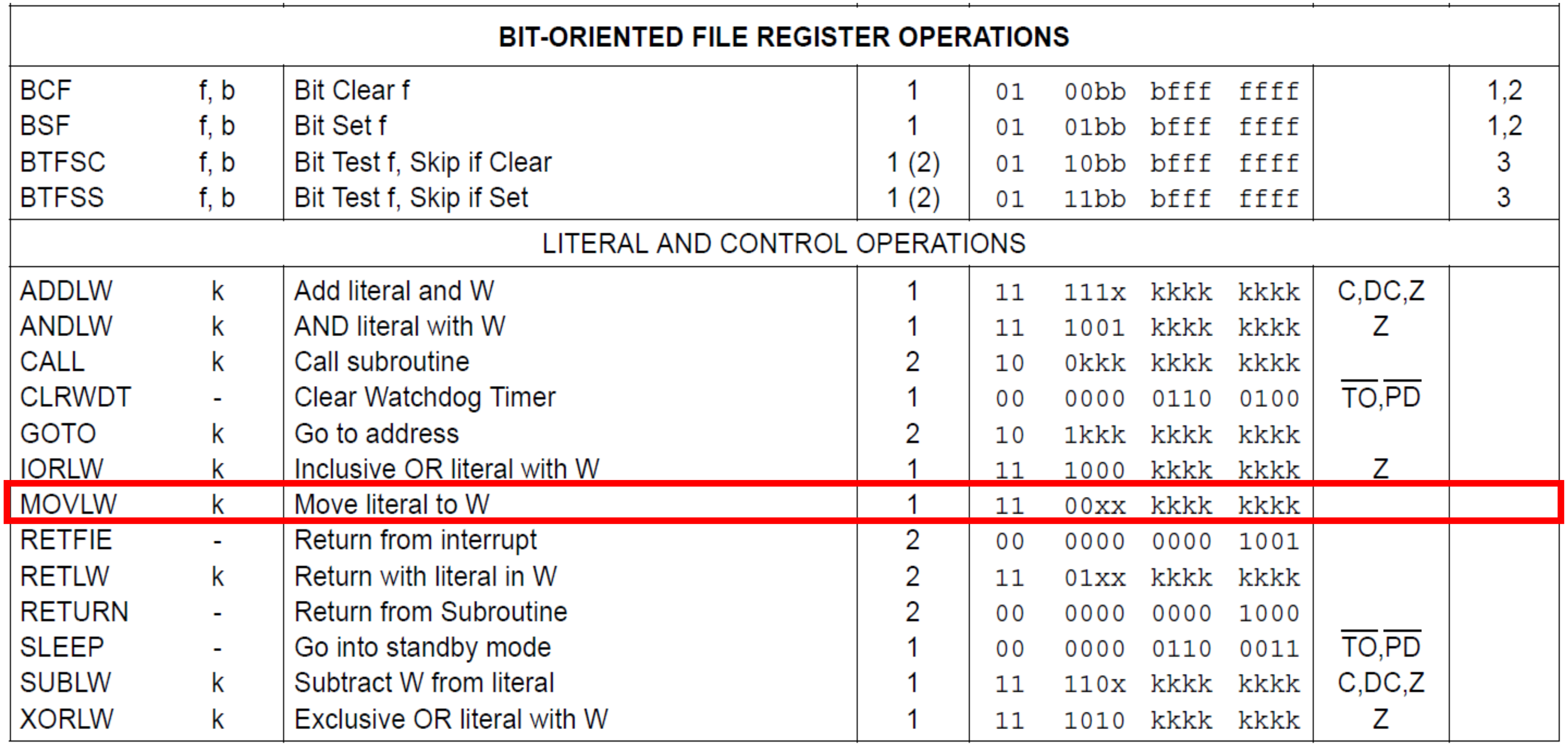


Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
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	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2

f: Data memoryのAddress
d: Data memory when d=0（f）
W register when d=1（W）

x: Don't care
C: Carry flag, DC: Digit Carry flag
Z：Zero flag
□ : Explained instructions in previous

Instruction Set of PIC16F84A (2)



BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
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	

f: Address of Data memory
b: Bit index for bit operation
k: Constant value (literal)

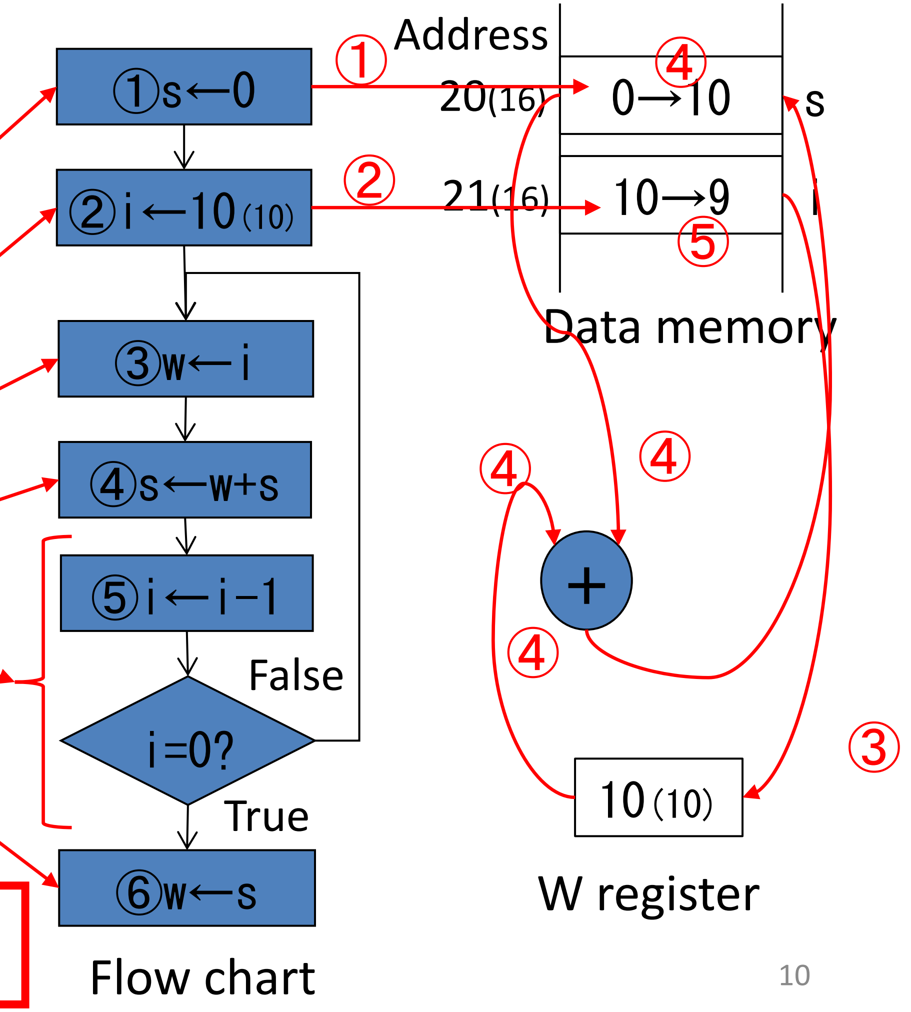
x: Don't care
C: Carry flag,   DC: Digit Carry flag
Z: Zero flag
TO, PD: Status for internal states of processor
□ : Explained instruction in previous

Summation program from 1 to 10

$$s = \sum_{i=1}^{10} i = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55$$

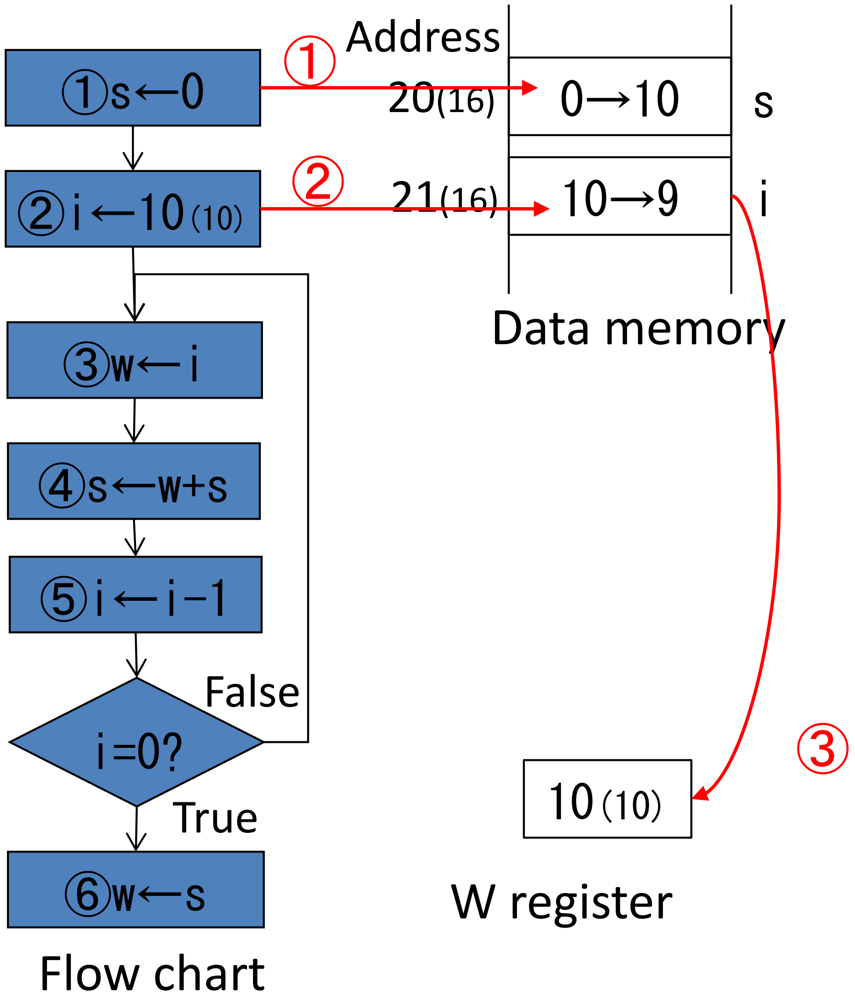
```
        INCLUDE ”p16f84a.inc”
        LIST     P=PIC16F84A
        ORG     0
        clrf    H'20'       ; s←0
        movlw   D'10'       ; w←10
        movwf   H'21'       ; i←10
LOOP    movf    H'21',w     ; w←i
        addwf   H'20',f     ; s←i+s
        decfsz  H'21',f     ; i←i-1
        goto    LOOP
        movf    H'20',w     ; w←s
        clrw
        end
```

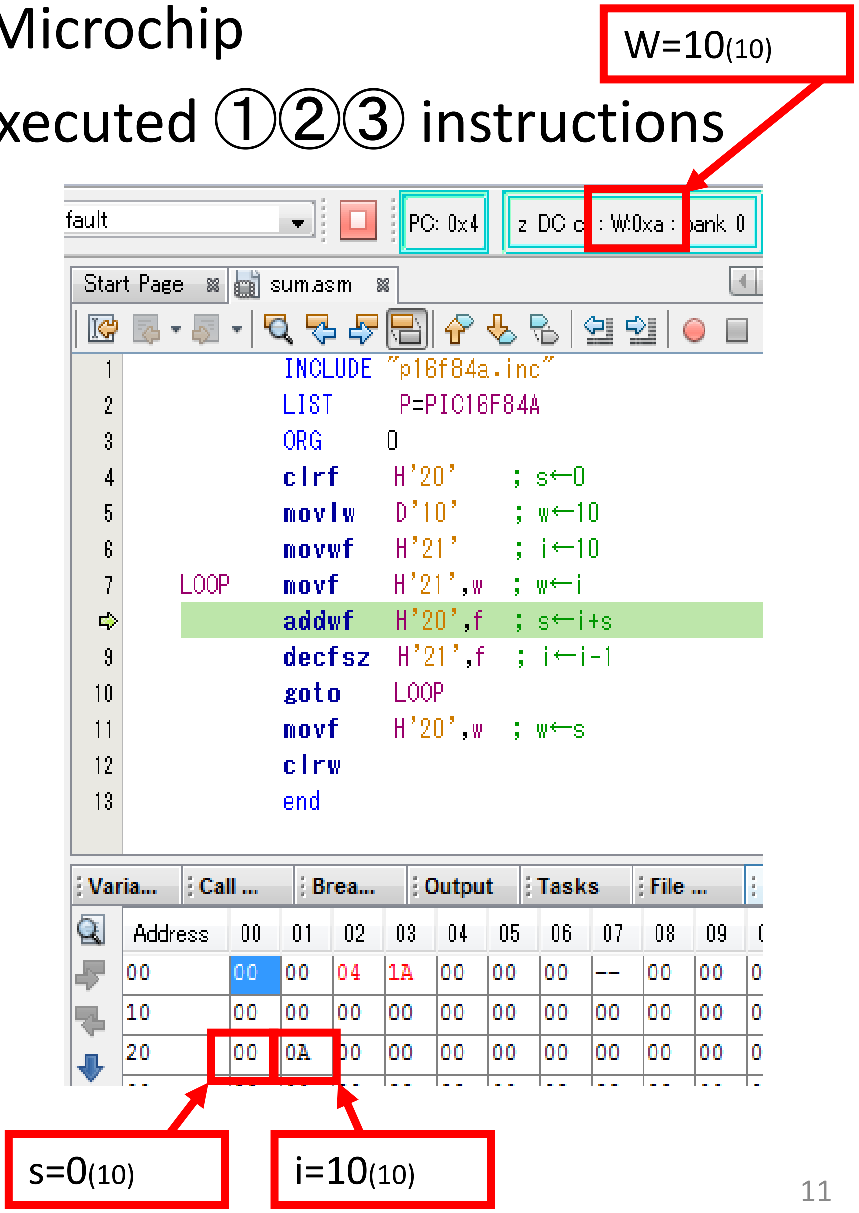
When i-1 equals 0, next instruction is canceled and after the next instruction will be executed.



Simulation Result (1)

- Simulator: MPLAB X IDE by Microchip
  - The first iteration after executed ①②③ instructions



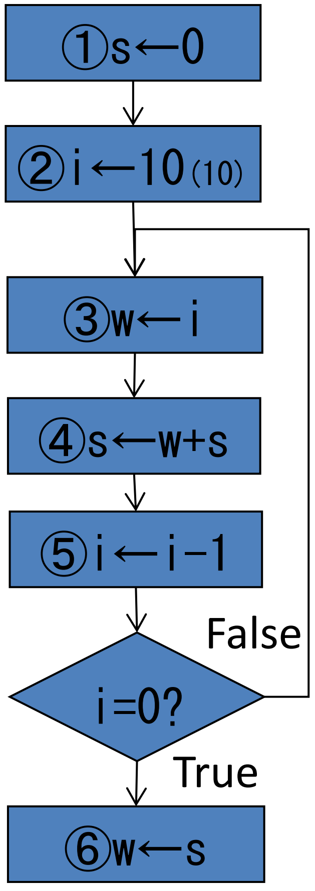


```
        INCLUDE "p16f84a.inc"
        LIST    P=PIC16F84A
        ORG     0
        clrf    H'20'   ; s←0
        movlw   D'10'   ; w←10
        movwf   H'21'   ; i←10
LOOP    movf    H'21',w ; w←i
        addwf   H'20',f ; s←i+s
        decfsz  H'21',f ; i←i-1
        goto    LOOP
        movf    H'20',w ; w←s
        clrw
        end
```

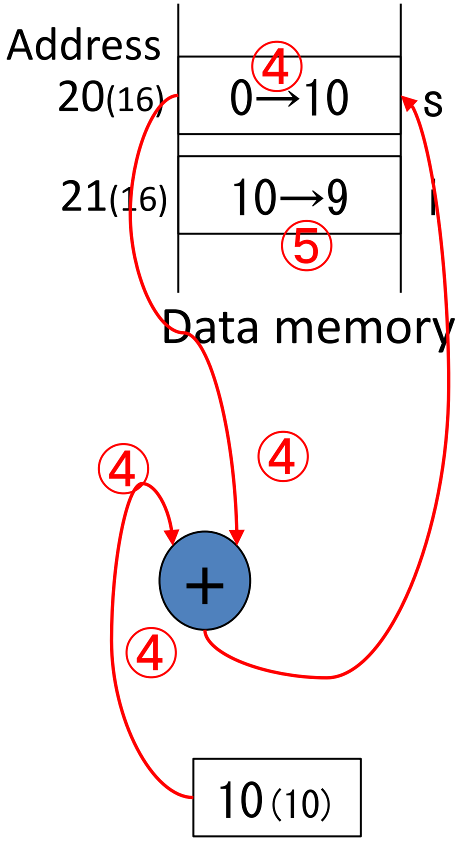
```
Address  00  01  02  03  04  05  06  07  08  09
00       00  00  04  1A  00  00  00  --  00  00
10       00  00  00  00  00  00  00  00  00  00
20       00  0A  00  00  00  00  00  00  00  00
```

Simulation Result (2)

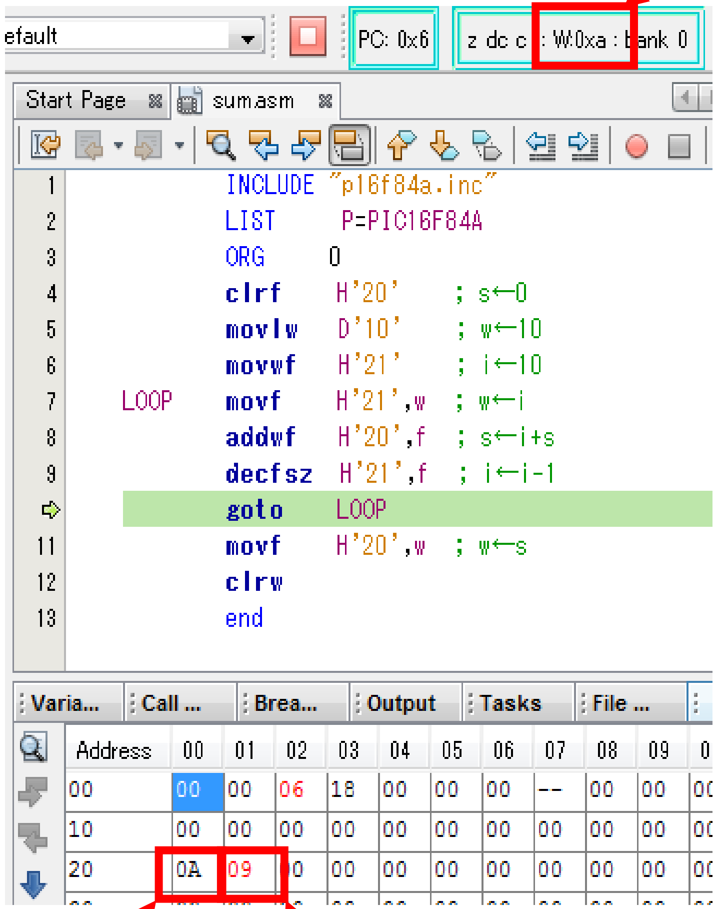
- Simulator: MPLAB X IDE by Microchip
  - The first iteration after executed ④⑤ instructions



Flow chart



W register



```
            INCLUDE "p16f84a.inc"
            LIST     P=PIC16F84A
            ORG     0
            clrf    H'20'     ; s←0
            movlw   D'10'     ; w←10
            movwf   H'21'     ; i←10
    LOOP    movf    H'21',w   ; w←i
            addwf   H'20',f   ; s←i+s
            decfsz  H'21',f   ; i←i-1
            goto    LOOP
            movf    H'20',w   ; w←s
            clrw
            end
```

```
Address  00  01  02  03  04  05  06  07  08  09  0
00       00  00  06  18  00  00  00  --  00  00  00
10       00  00  00  00  00  00  00  00  00  00  00
20       0A  09  00  00  00  00  00  00  00  00  00
```

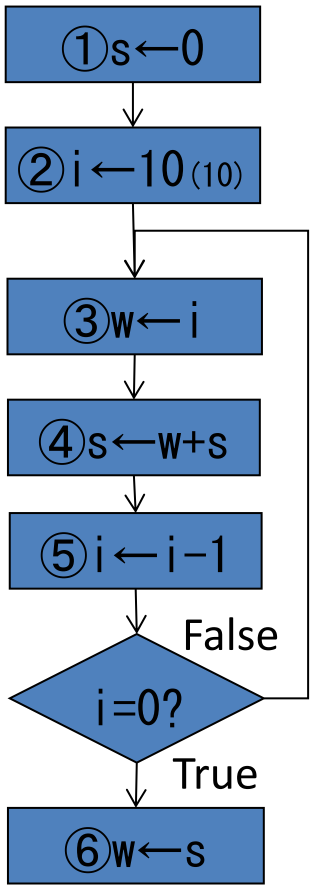
W=10(10)

s=10(10)

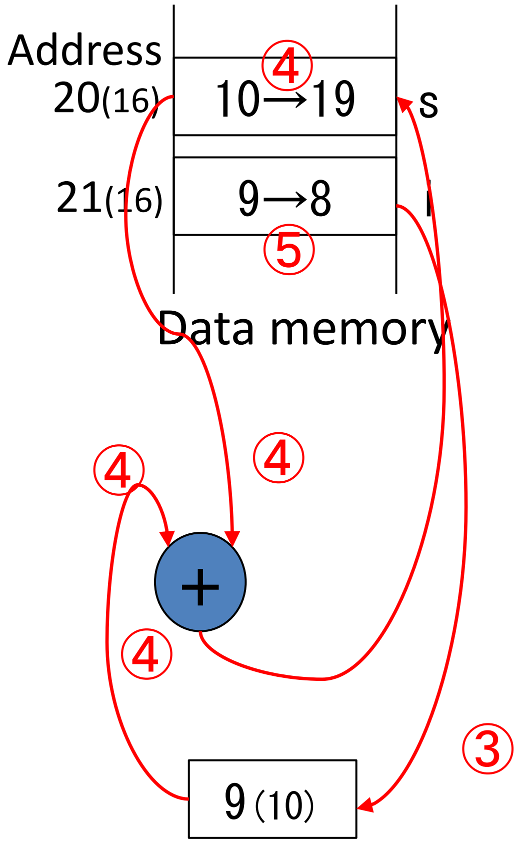
i=9(10)

Simulation Result (3)

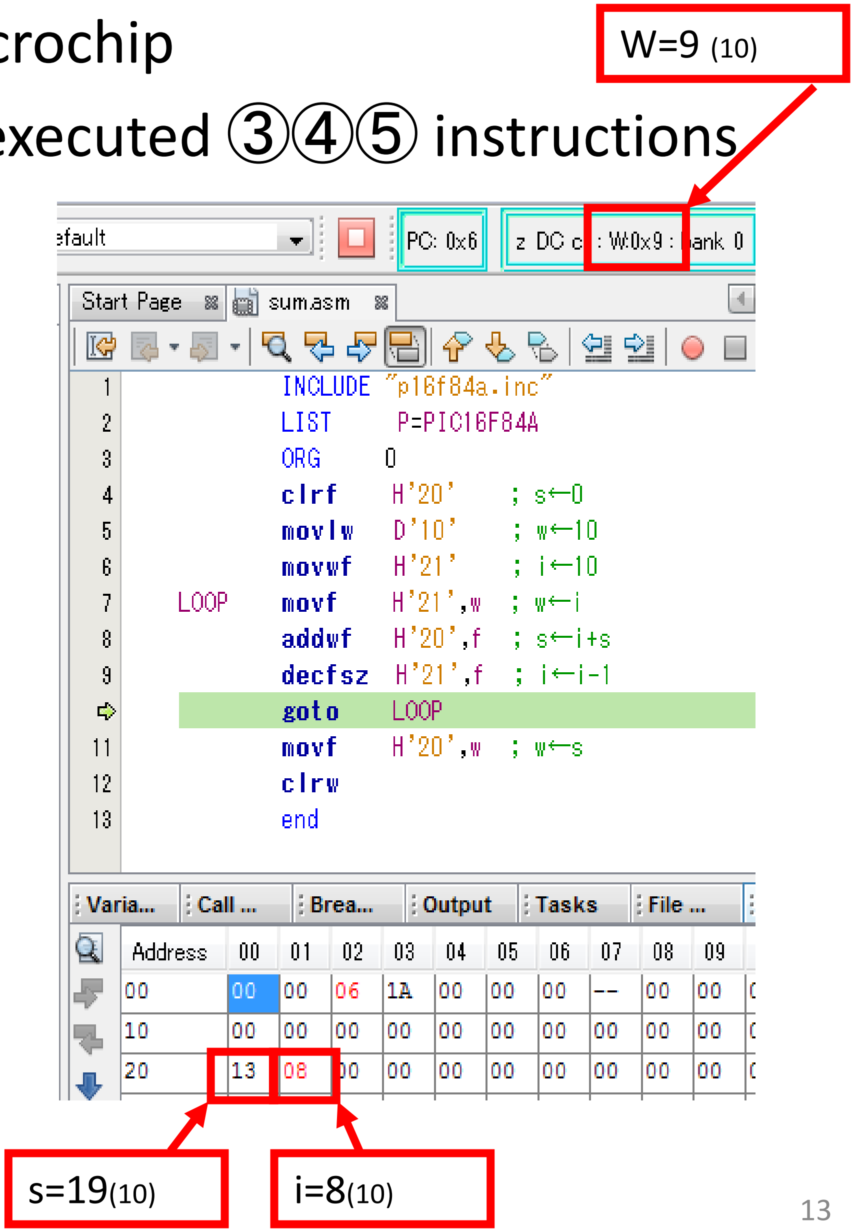
- Simulator: MPLAB X IDE by Microchip
  - The second iteration after executed ③④⑤ instructions



Flow chart



W register

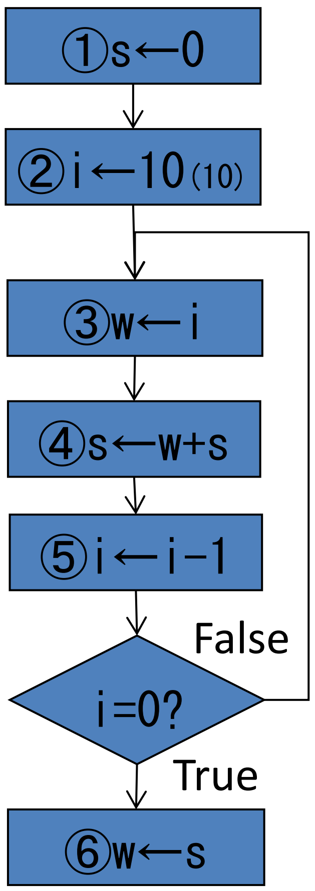


```
        INCLUDE "p16f84a.inc"
        LIST     P=PIC16F84A
        ORG      0
        clrf     H'20'      ; s←0
        movlw    D'10'      ; w←10
        movwf    H'21'      ; i←10
LOOP    movf     H'21',w    ; w←i
        addwf    H'20',f    ; s←i+s
        decfsz   H'21',f    ; i←i-1
        goto     LOOP
        movf     H'20',w    ; w←s
        clrw
        end
```

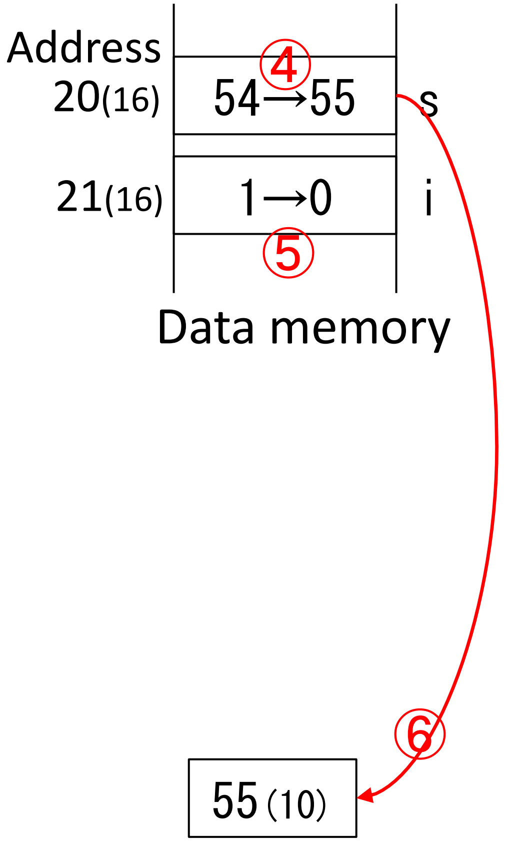
```
Address  00  01  02  03  04  05  06  07  08  09
00       00  00  06  1A  00  00  00  --  00  00
10       00  00  00  00  00  00  00  00  00  00
20       13  08  00  00  00  00  00  00  00  00
```

Simulation Result (4)

- Simulator: MPLAB X IDE by Microchip
  - The tenth iteration after executed ④⑤⑥ instructions

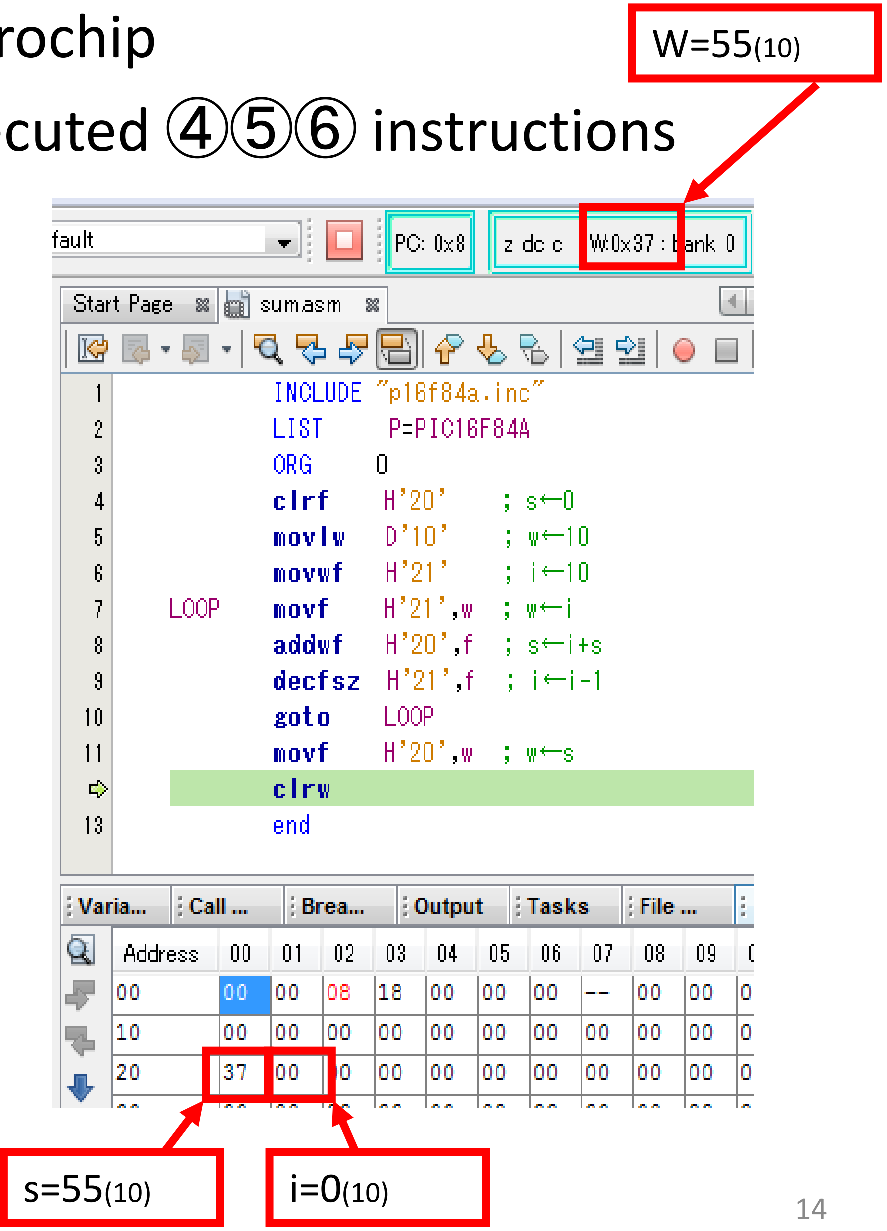


Flow chart



Data memory

W register



W=55(10)

```
            INCLUDE "p16f84a.inc"
            LIST     P=PIC16F84A
            ORG      0
            clrf     H'20'     ; s←0
            movlw    D'10'     ; w←10
            movwf    H'21'     ; i←10
LOOP        movf     H'21',w   ; w←i
            addwf    H'20',f   ; s←i+s
            decfsz   H'21',f   ; i←i-1
            goto     LOOP
            movf     H'20',w   ; w←s
            clrw
            end
```

Address	00	01	02	03	04	05	06	07	08	09
00	00	00	08	18	00	00	00	--	00	00
10	00	00	00	00	00	00	00	00	00	00
20	37	00	0	00	00	00	00	00	00	00

s=55(10)

i=0(10)

Differences from the original

Original PIC16

- 4 cycles in one stage
- Selectable external clock or internal clock
- Sleep mode (Low power consumption mode)
- Flash memory

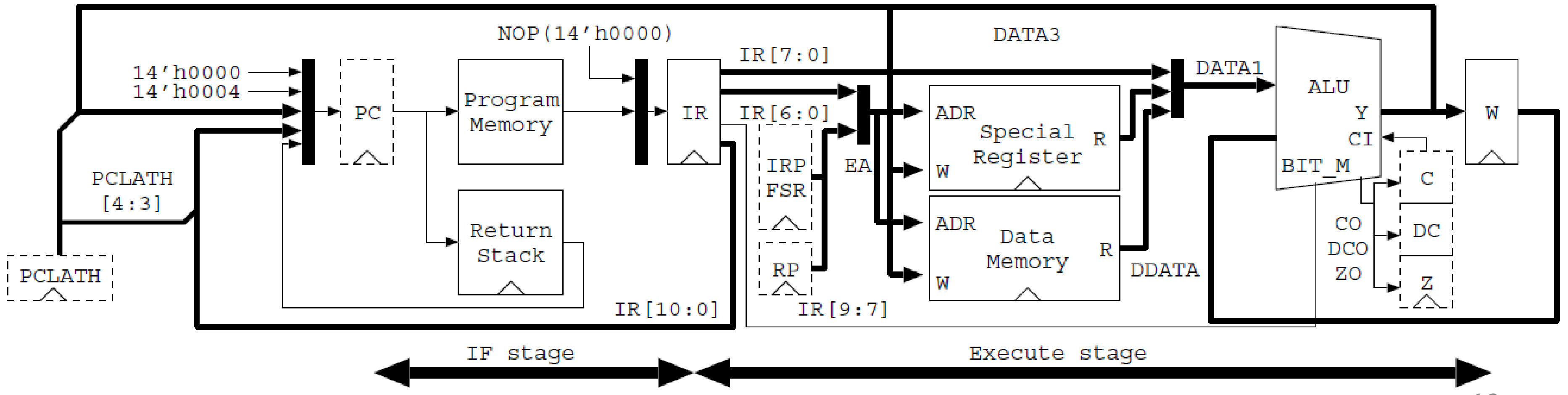
Our specification

- 1 cycle in one stage
- Only external clock
- Different behavior by 1 cycle in one stage:
  interrupt and I/O timing
- Pseudo Sleep mode (It is repeating NOP instruction)
- No flash memory

Block Diagram

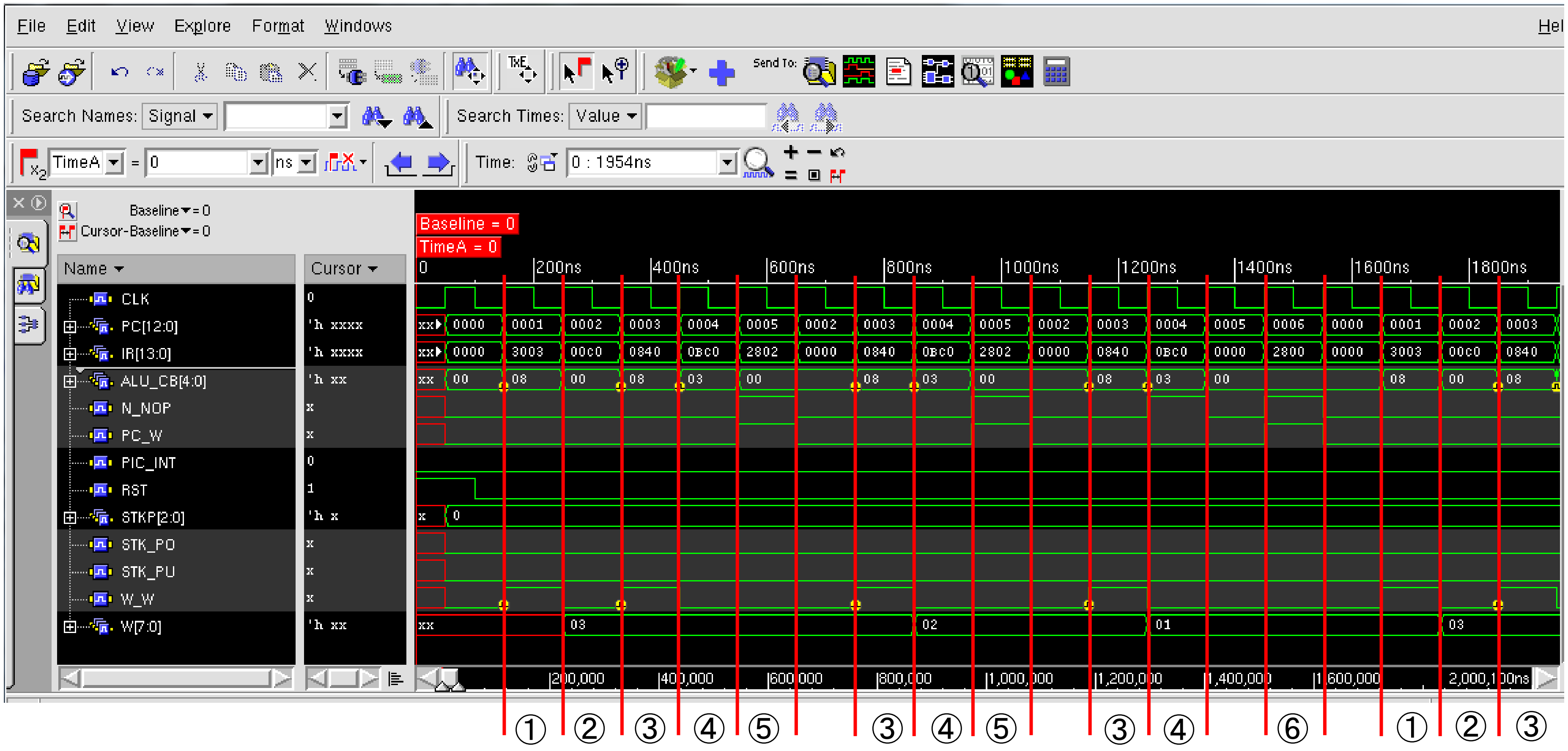
- 2-stage Pipeline
  - Instruction Fetch (IF) stage
  - Execution (E) stage

  } Each stage is executed in parallel
- Critical Path
  - IR→Data Memory→ALU→Data Memory
  - Operating frequency cannot be faster because of Read/Write accessing of Data Memory in one cycle.



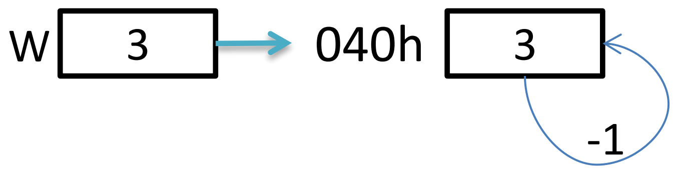
Behavior of 2 Stage Pipeline Processor

Behavior of PIC16 compatible processor designed in our Lab. （Note that original PIC 16 has 4 cycles in a one stage）



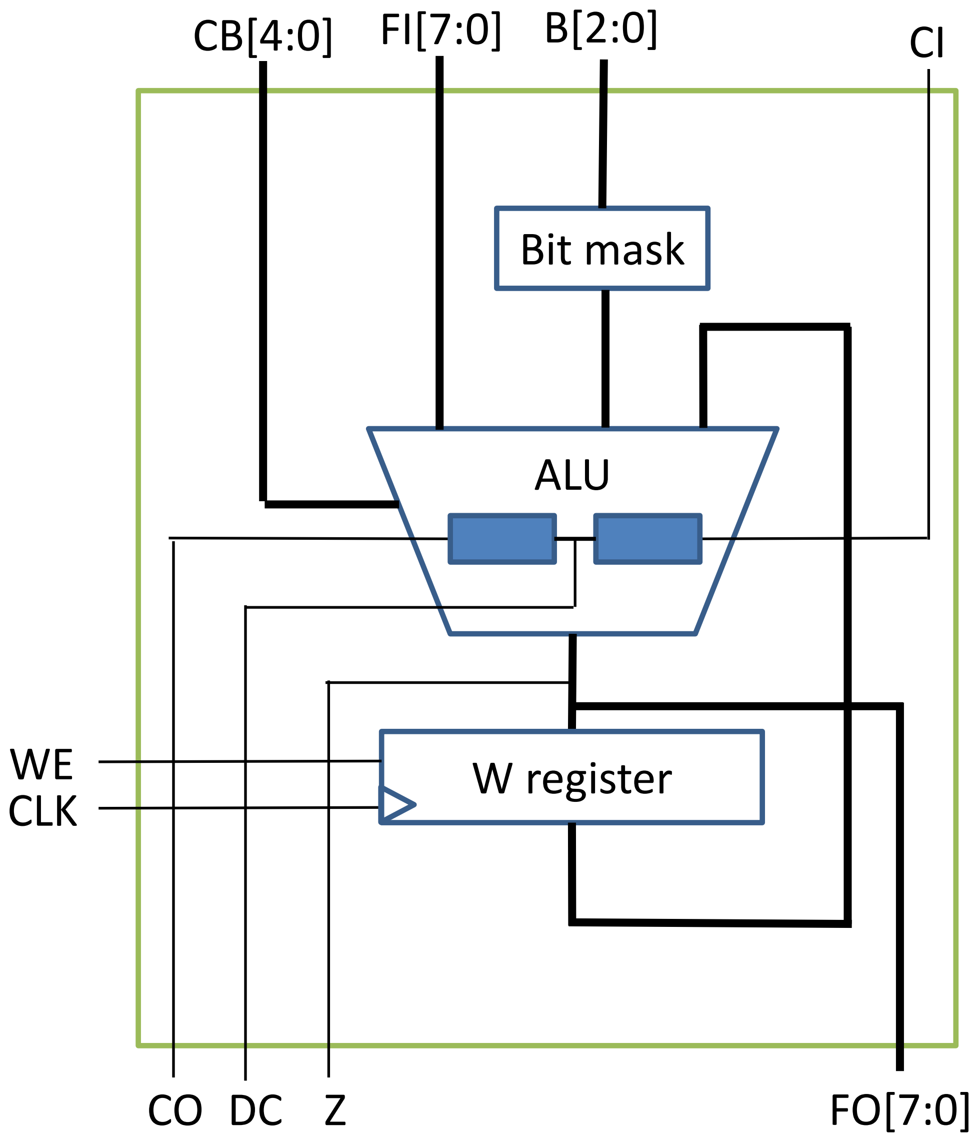
File Edit View Explore Format Windows Help
Search Names: Signal
Search Times: Value
TimeA = 0 ns
Time: 0 : 1954ns
Baseline = 0
Cursor-Baseline = 0
Baseline = 0
TimeA = 0
Name
Cursor
CLK 0
PC[12:0] 'h xxxx
IR[13:0] 'h xxxx
ALU_CB[4:0] 'h xx
N_NOP x
PC_W x
PIC_INT 0
RST 1
STKP[2:0] 'h x
STK_PO x
STK_PU x
W_W x
W[7:0] 'h xx
0 200ns 400ns 600ns 800ns 1000ns 1200ns 1400ns 1600ns 1800ns
xx 0000 0001 0002 0003 0004 0005 0002 0003 0004 0005 0002 0003 0004 0005 0006 0000 0001 0002 0003
xx 0000 3003 00C0 0840 0BC0 2802 0000 0840 0BC0 2802 0000 0840 0BC0 0000 2800 0000 3003 00C0 0840
xx 00 08 00 08 03 00 08 03 00 08 03 00 08 00 08
x 0
xx 03 02 01 03
200,000 400,000 600,000 800,000 1,000,000 1,200,000 1,400,000 1,600,000 2,000,100ns
① ② ③ ④ ⑤ ③ ④ ⑤ ③ ④ ⑥ ① ② ③

```
0000              00001          ORG    0
0000    3003      00002 L2:    ① MOVLW  3
0001    00C0      00003        ② MOVWF  040h
0002    0840      00004 L1:    ③ MOVF   040h, 0
0003    0BC0      00005        ④ DECFSZ 040h, 1
0004    2802      00006        ⑤ GOTO   L1
0005    2800      00007        ⑥ GOTO   L2
                  00008          end
```



DESIGN EXAMPLE

ALU Design for PIC16



CLK	Clock
CB[4:0]	operation code
WE	Write enable for W reg.
B[2:0]	bit position
FI[7:0]	operand
FO[7:0]	Result data
CI	Carry in
CO	Carry out
DC	Half carry for ADD/SUB
Z	Zero flag

alu.v : module alu

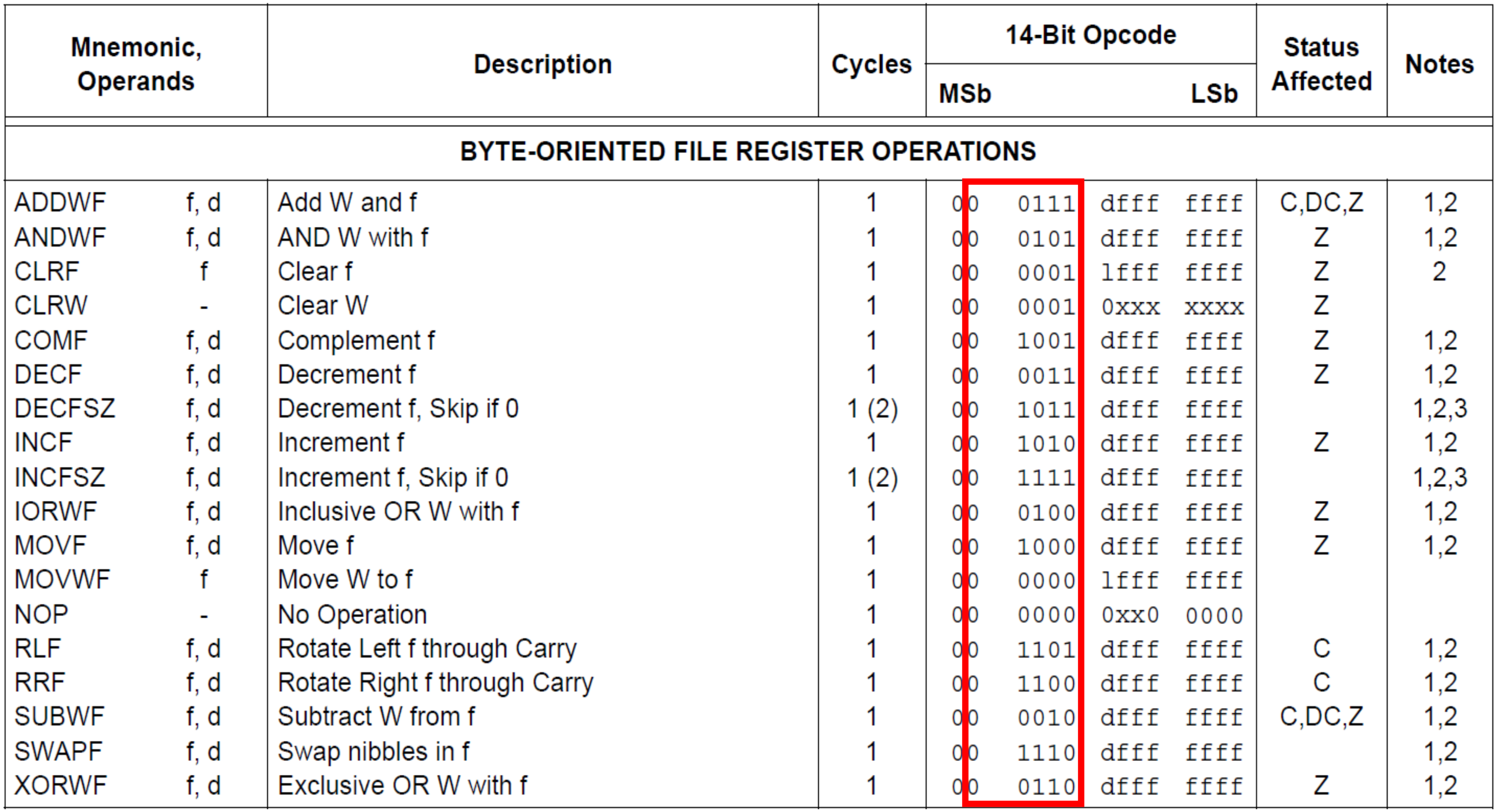
Refer the skeleton code, “alu.v”

```
//
// ALU for PIC16
//
`include "alu_op.v"

module alu ( CLK, CB, WE, B, FI, FO, CI, CO, DC, Z );
   input        CLK;// Clock
   input  [4:0] CB; // operation code
   input        WE; // Write enable for W register
   input  [2:0] B;  // bit position
   input  [7:0] FI; // left operand
   output [7:0] FO; // result data
   input        CI; // Carry in                 ______
   output       CO; // Carry out (ADD:Carry/SUB:Borrow)
   output       DC; // Half carry
   output       Z;  // Zero

   ...

endmodule
```

alu_op.v : ALU operation code

```
// Control code for PIC16 ALU
`define      IPSW   5'b00000 // Pass W
`define      ICLR   5'b00001 // Clear F and W
`define      ISUB   5'b00010 // Arithmetic Subtract
`define      IDEC1  5'b00011 // Decrement for DECF
`define      IOR    5'b00100 // Logical OR
`define      IAND   5'b00101 // Logical AND
`define      IXOR   5'b00110 // Logical exclusive OR
`define      IADD   5'b00111 // Arithmetic Add
`define      IPSF   5'b01000 // Pass F
`define      INTF   5'b01001 // Logical Complement F (NOT)
`define      IINC1  5'b01010 // Increment for INCF
`define      IDEC2  5'b01011 // Decrement for DECFSZ
`define      IRRF   5'b01100 // Rotate Right F with carry
`define      IRLF   5'b01101 // Rotate Left F with carry
`define      ISWP   5'b01110 // Nibble swap F
`define      IINC2  5'b01111 // Increment for INCFSZ
`define      IBCF   5'b100?? // Bit Clear F
`define      IBSF   5'b101?? // Bit Set F
`define      IBTF   5'b11??? // Bit Test F
```

Instruction Set of PIC16F84A (1)



Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
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	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2

f: Data memoryのAddress
d: Data memory when d=0（f）
W register when d=1（W）

x: Don't care
C: Carry flag,　DC: Digit Carry flag
Z：Zero flag
🟥 : ALU operation code

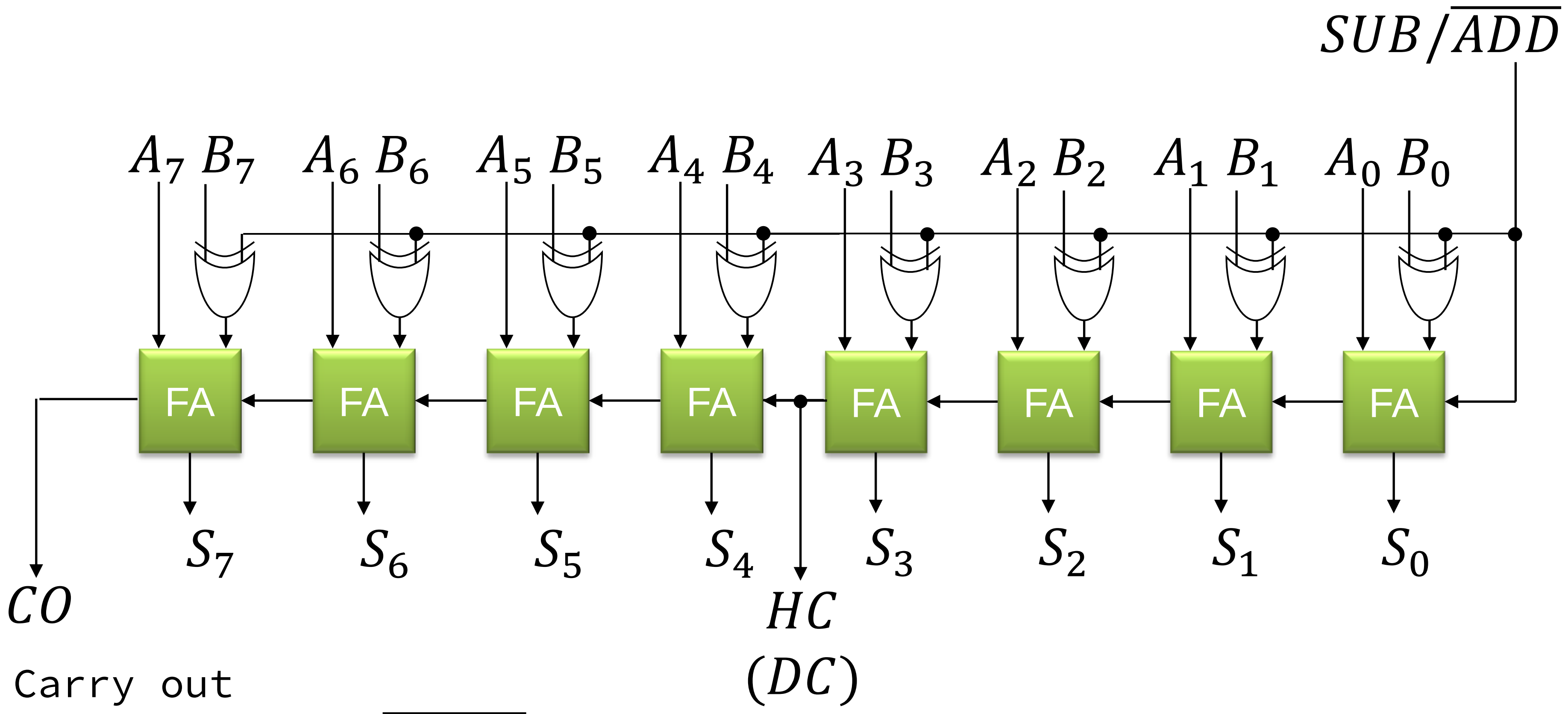
Instruction Set of PIC16F84A (2)

BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit Clear f	1	01 00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb	bfff	ffff		3
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	

f: Address of Data memory
b: Bit index for bit operation
k: Constant value (literal)

x: Don't care
C: Carry flag, DC: Digit Carry flag
Z : Zero flag
TO, PD: Status for internal states of processor
☐ : ALU operation code

ADD/SUB unit for PIC ALU



$SUB/\overline{ADD}$
A_7 B_7 A_6 B_6 A_5 B_5 A_4 B_4 A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
FA
FA
FA
FA
FA
FA
FA
FA
S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0
CO
HC
(DC)

Carry out
(ADD:Carry/SUB:$\overline{\text{Borrow}}$)

To be continued…