Try to Design and Implementation of a PIC16 compatible microcontroller

Skill up course

PIC16 Microcontroller

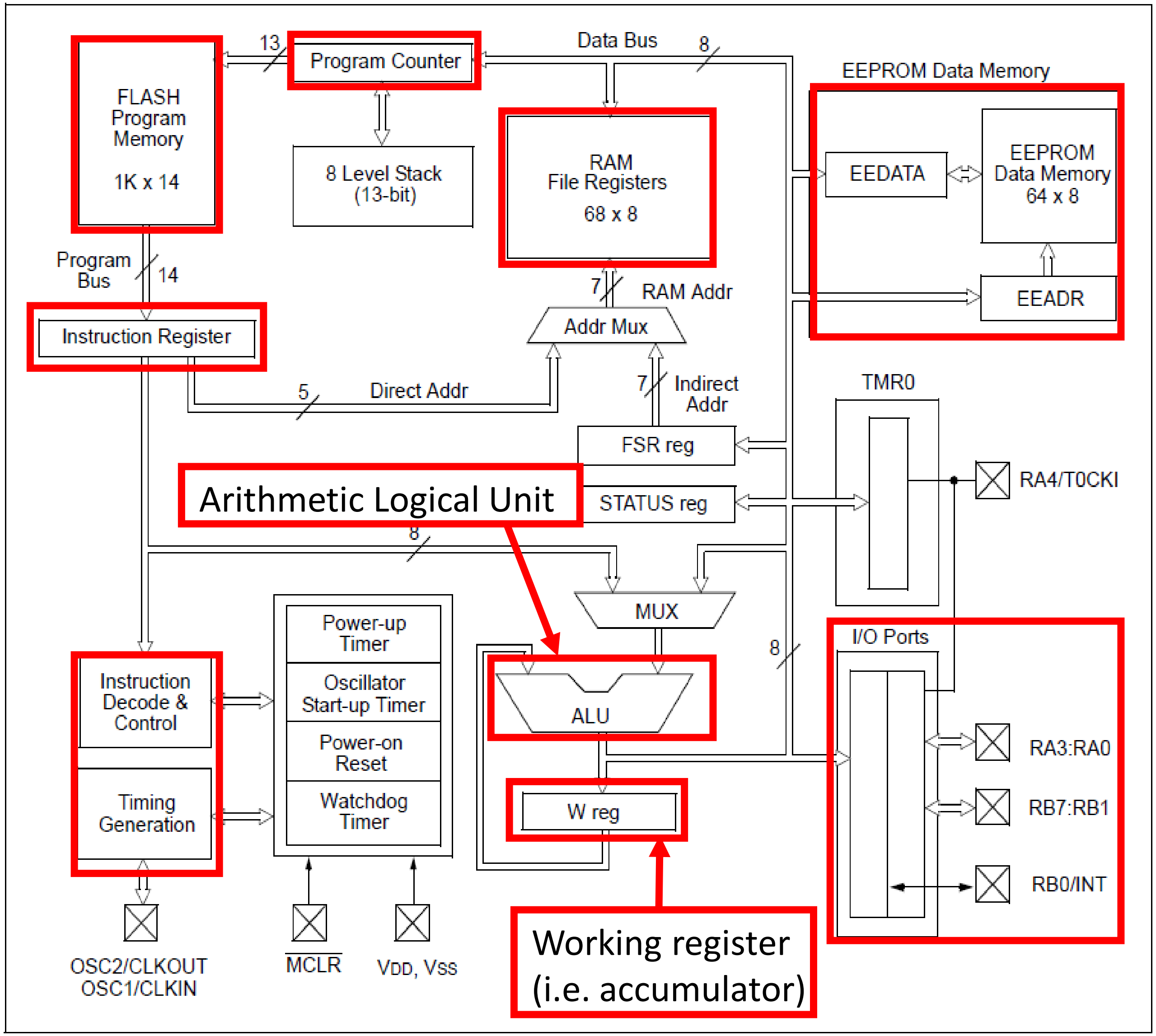
PIC16F84A

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



Arithmetic Logical Unit

Working register (i.e. accumulator)

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

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

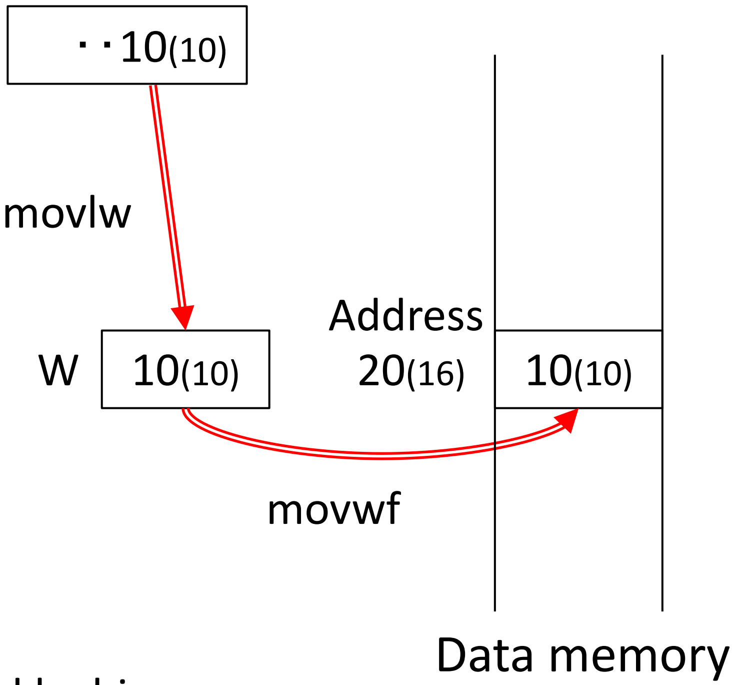
Assembly Language:
It is a language for ease of understanding machine language.

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.

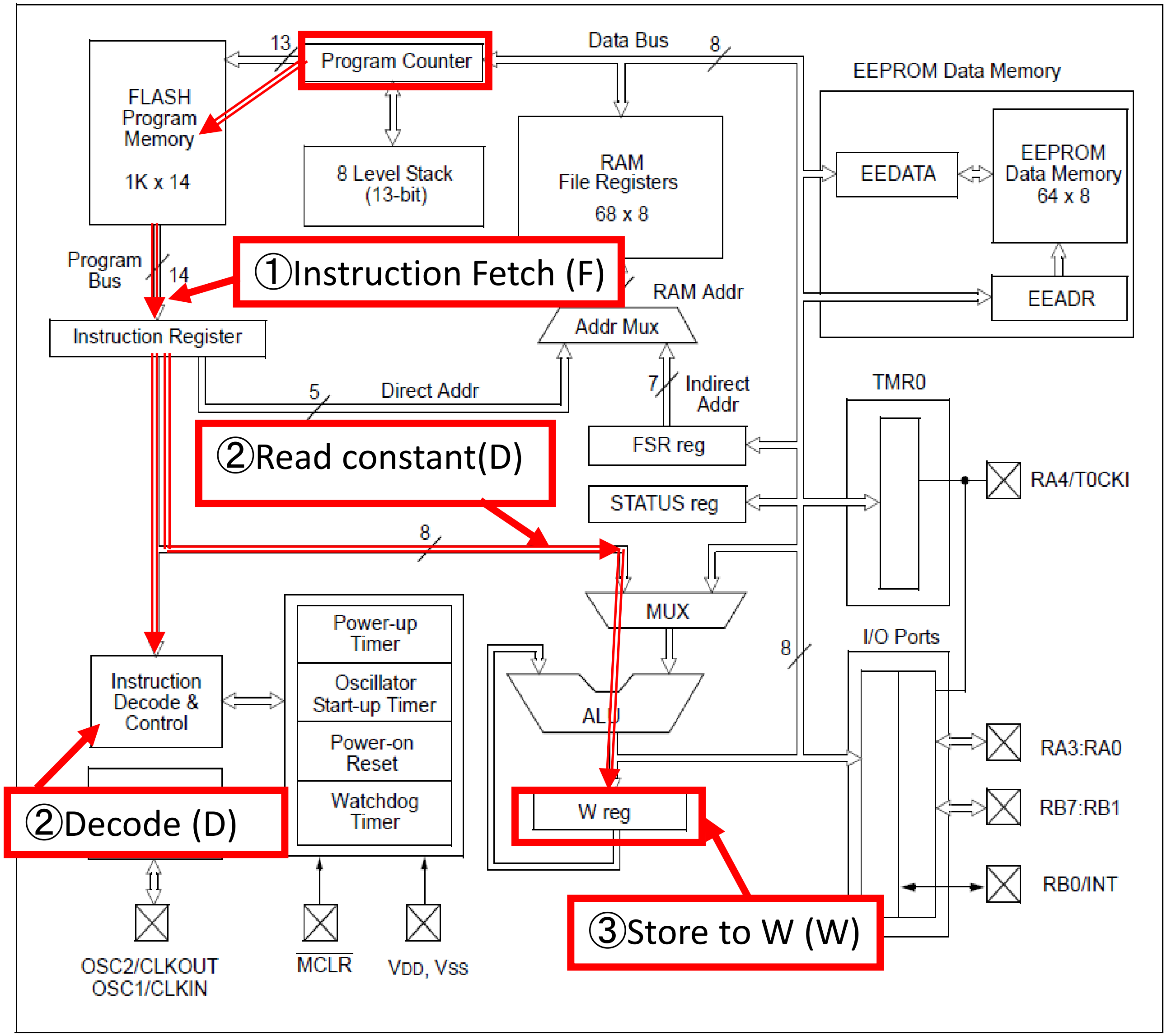
movlw constant

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



①Instruction Fetch (F)

②Read constant(D)

②Decode (D)

③Store to W (W)

Behavior of "movwf" instruction

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

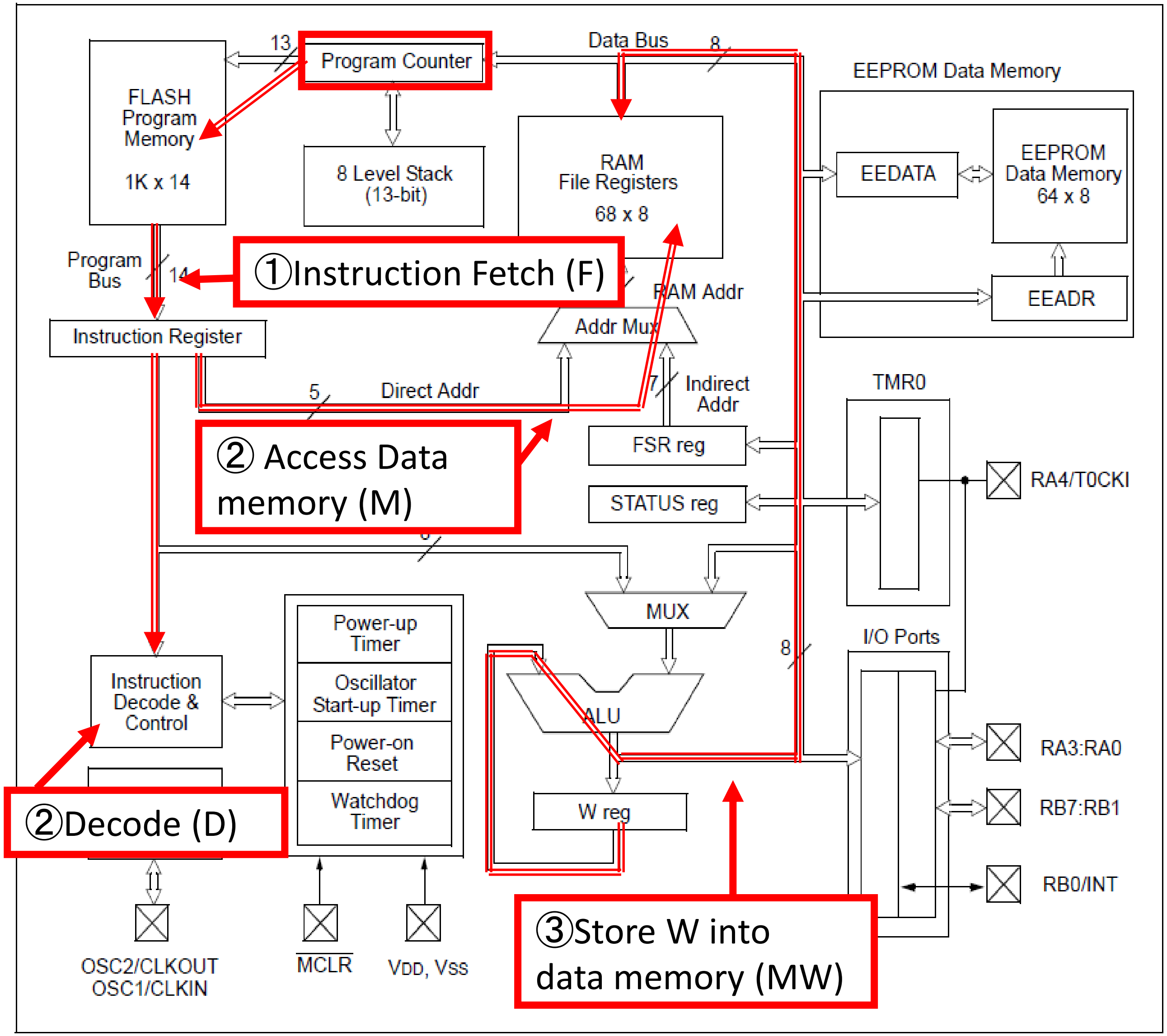
movwf Address

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



①Instruction Fetch (F)

② Access Data memory (M)

②Decode (D)

③Store W into data memory (MW)

Sample Program：Addition

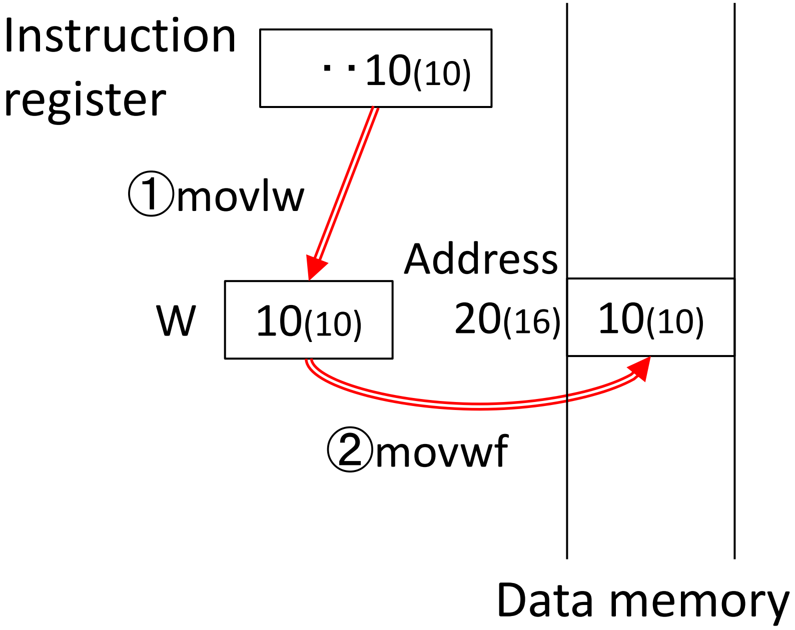
Add a value in data memory and W register by "addwf" instruction （Example: 10+20=30）

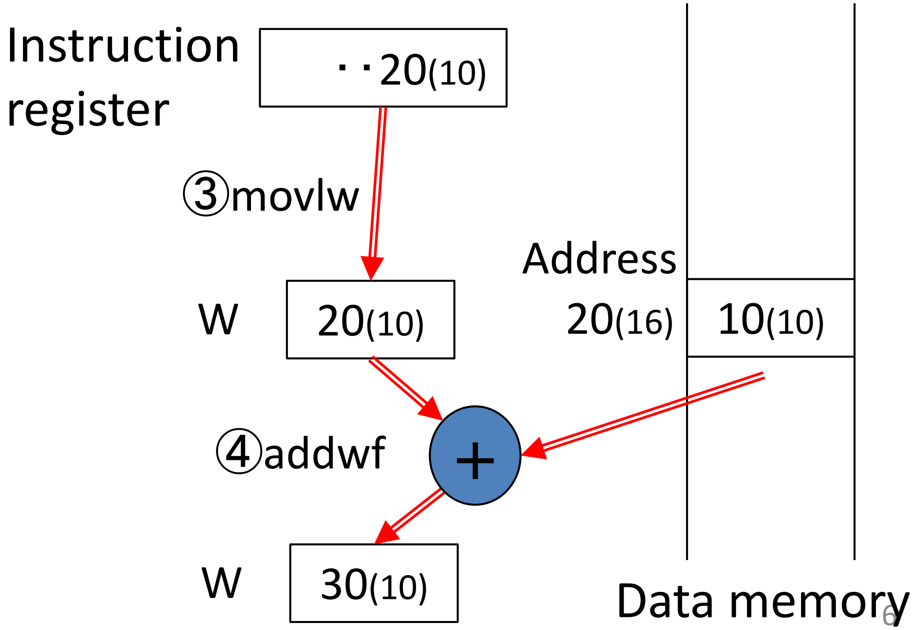
① movlw D'10' Store a constant $10_{(10)}$ in instruction field to W register

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.

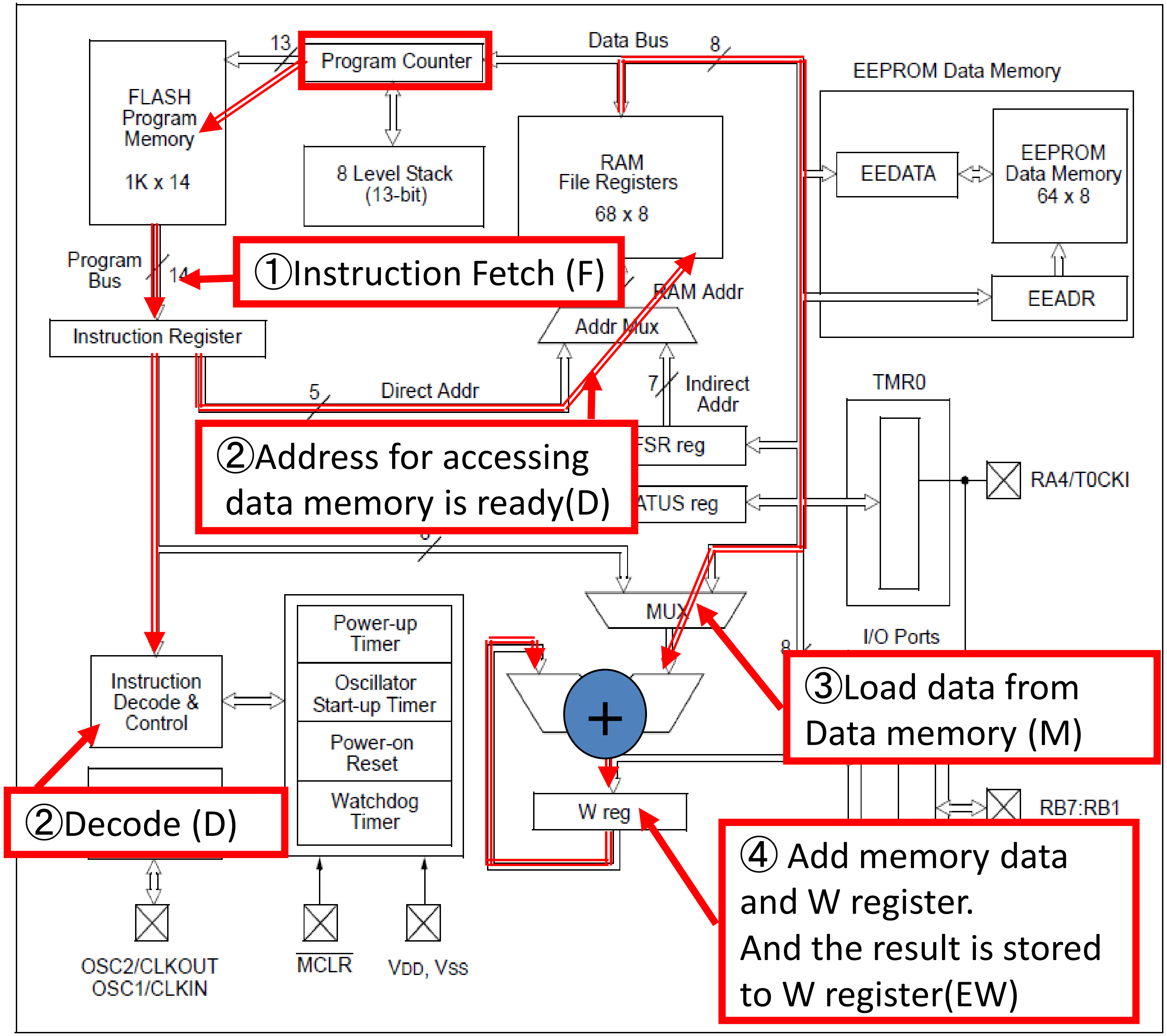
addwf Address, w

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



①Instruction Fetch (F)

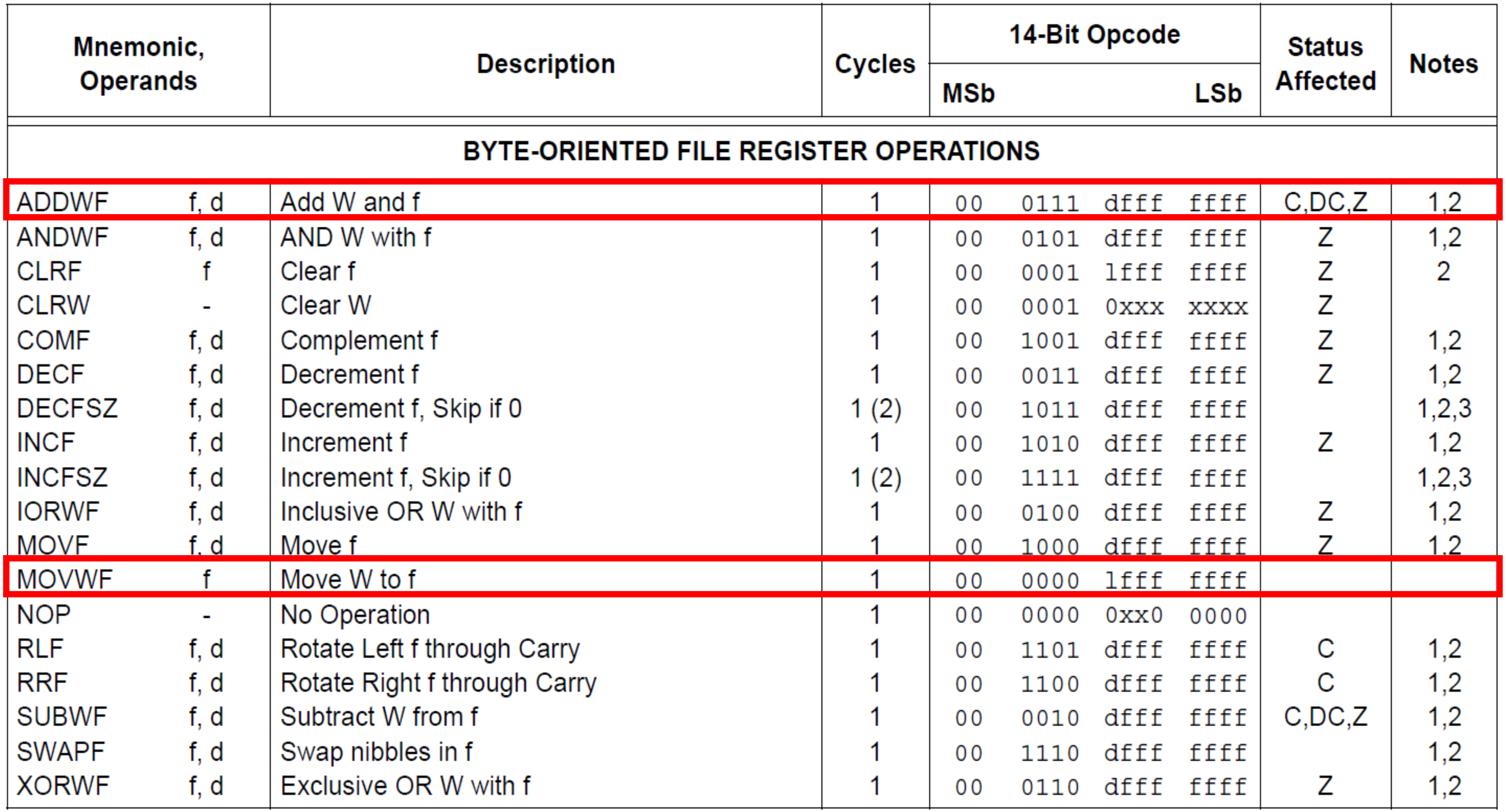
②Address for accessing data memory is ready(D)

②Decode (D)

③Load data from Data memory (M)

④ Add memory data and W register.
And the result is stored to W register(EW)

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）

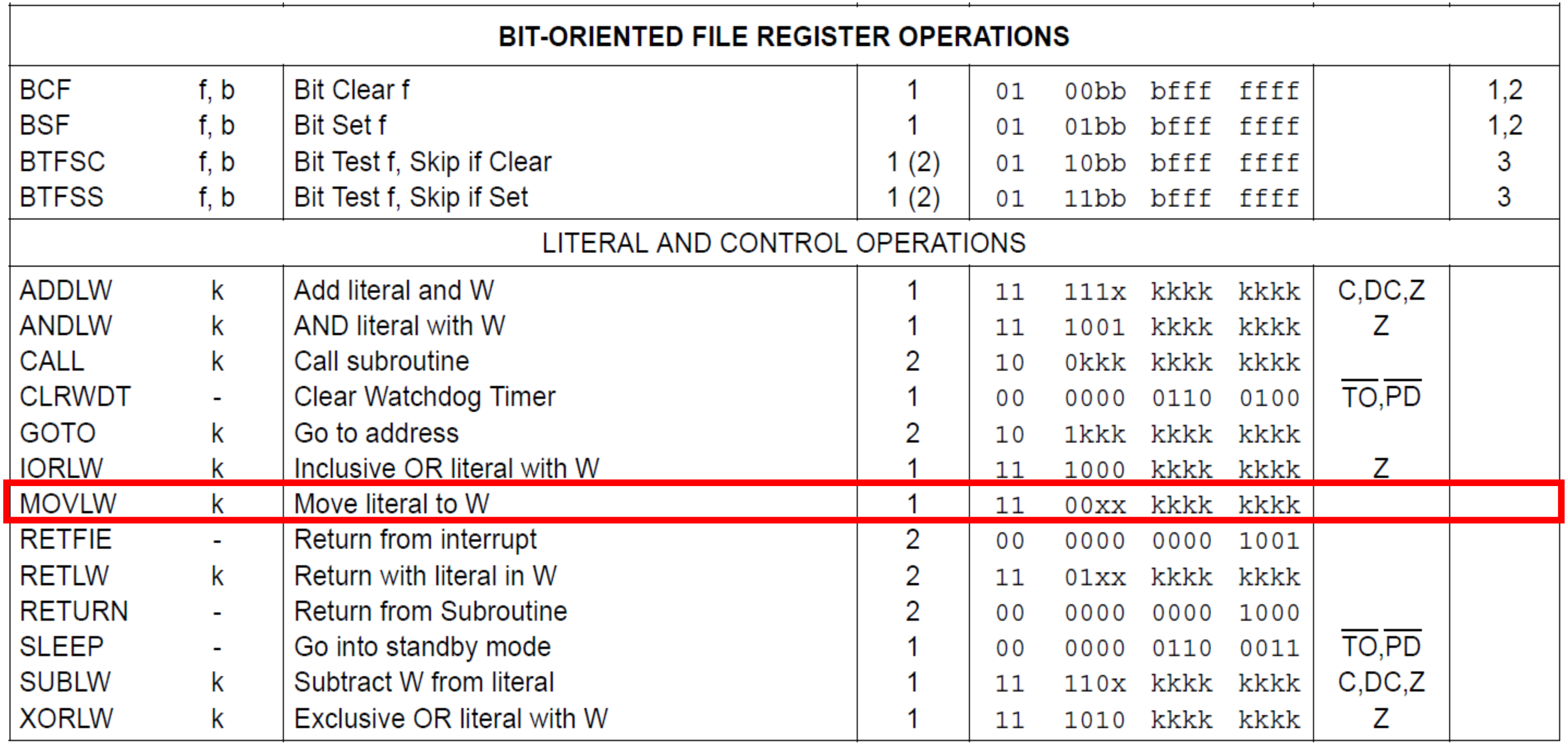
x: Don't care

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	$\overline{\text{TO}}$,$\overline{\text{PD}}$	
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	$\overline{\text{TO}}$,$\overline{\text{PD}}$	
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

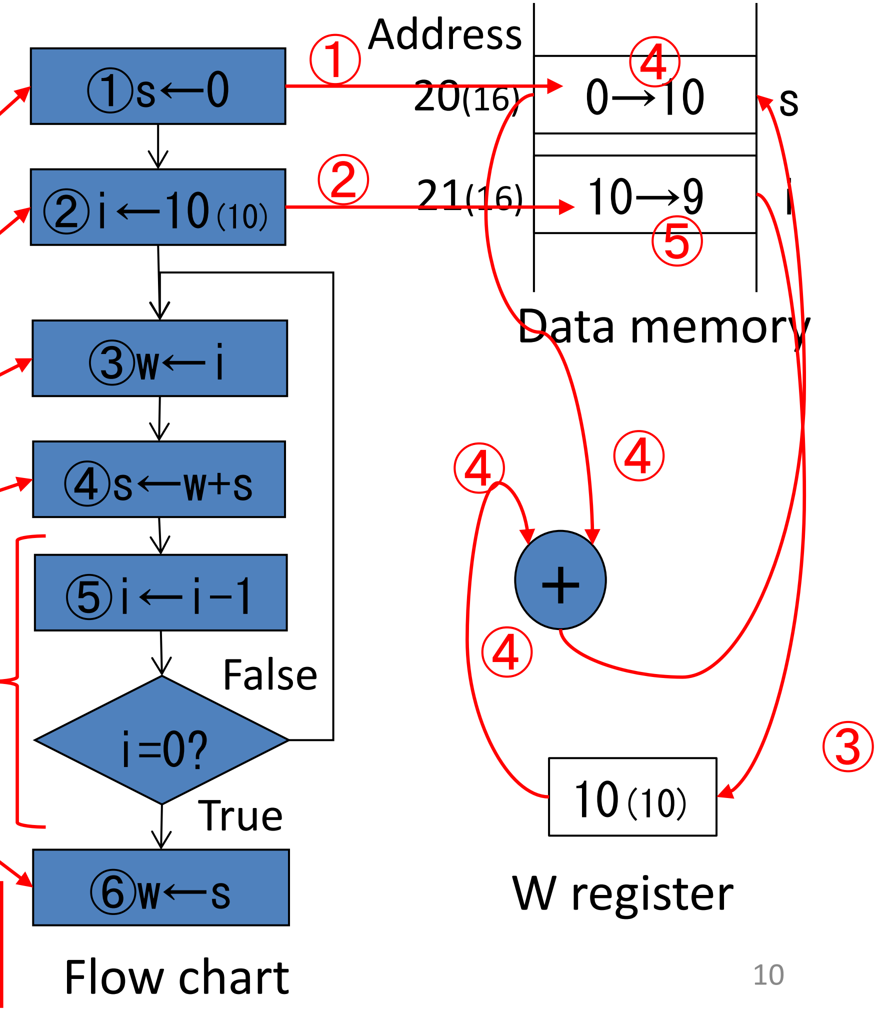
□ : Explained instruction in previous

Summation program from 1 to 10

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

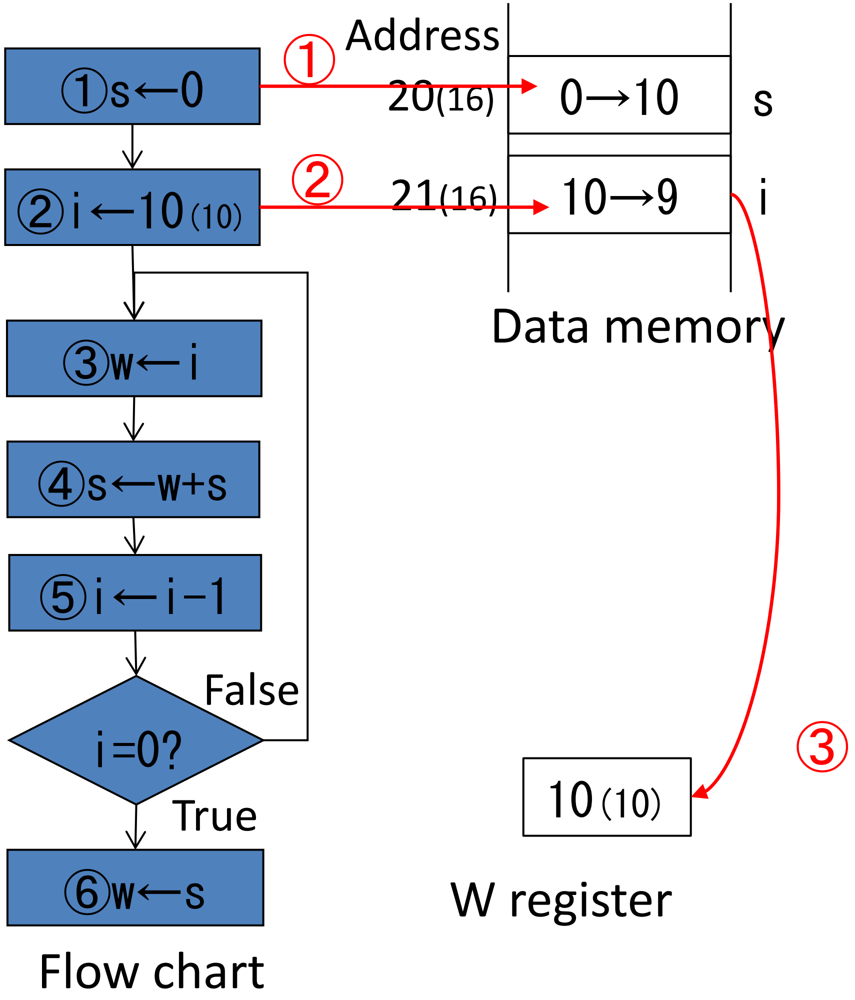
```
        INCLUDE ”p16f84a.inc”
        LIST     P=PIC16F84A
        ORG      0
        clrf     H'20'       ; s←0
        movlw    D'10'       ; w←10
        movwf    H'21'       ; i←10
LOOP    movf     H'21',w     ; w←i
        addwf    H'20',f     ; s←i+s
        decfsz   H'21',f     ; i←i-1
  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

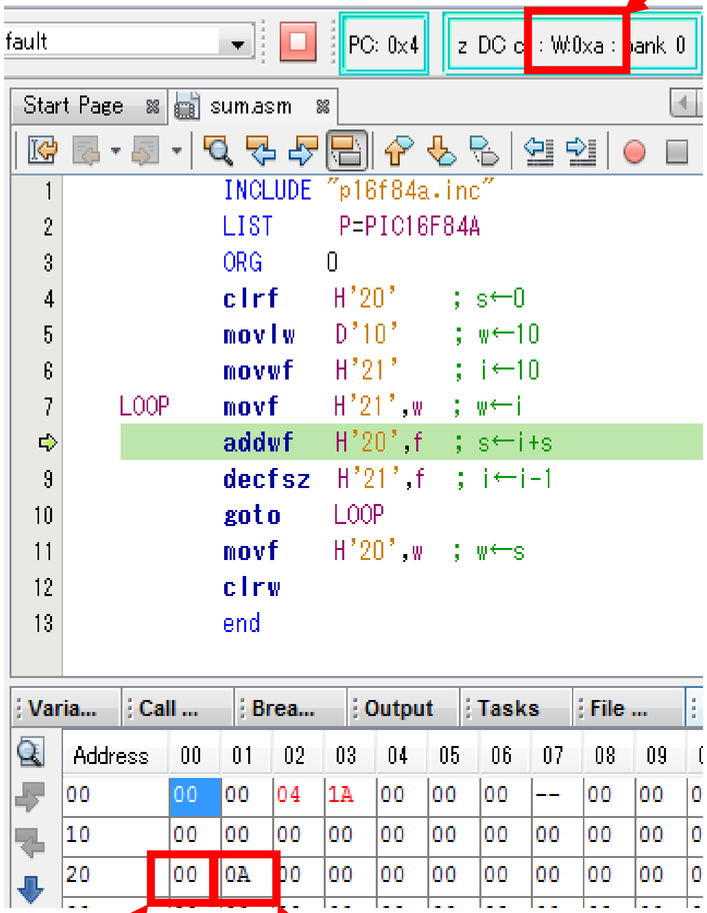


Simulation Result (1)

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



W=10(10)



PC: 0x4   z DC c : W:0xa : ank 0

```
            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
```

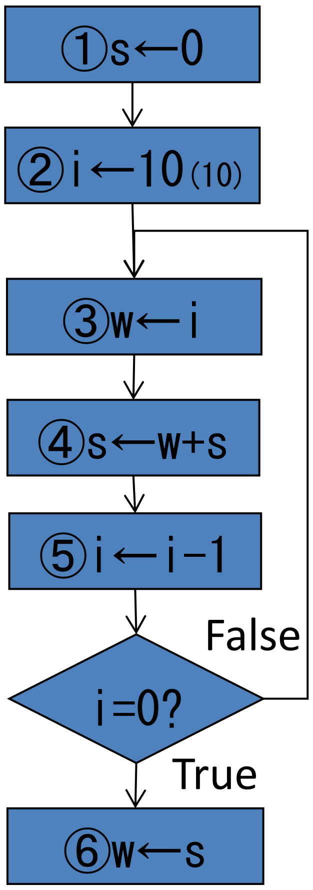
```
Address  00  01  02  03  04  05  06  07  08  09  0
00       00  00  04  1A  00  00  00  --  00  00  0
10       00  00  00  00  00  00  00  00  00  00  0
20       00  0A  00  00  00  00  00  00  00  00  0
```

s=0(10)

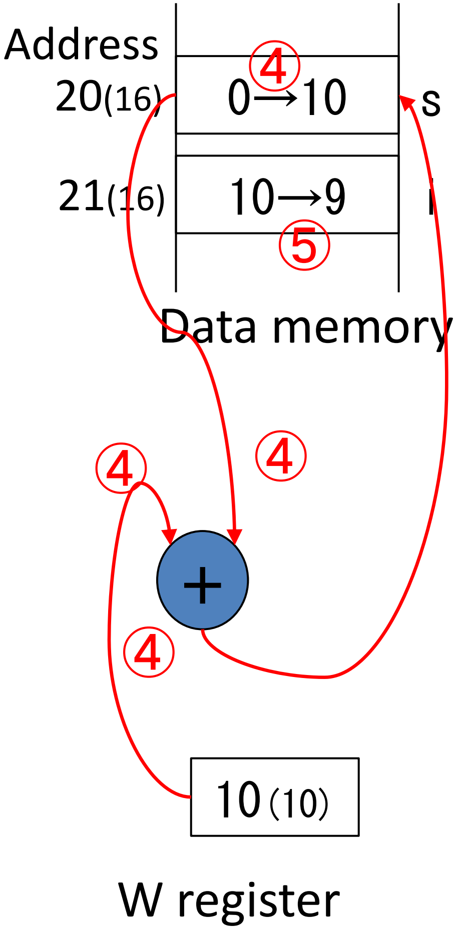
i=10(10)

Simulation Result (2)

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

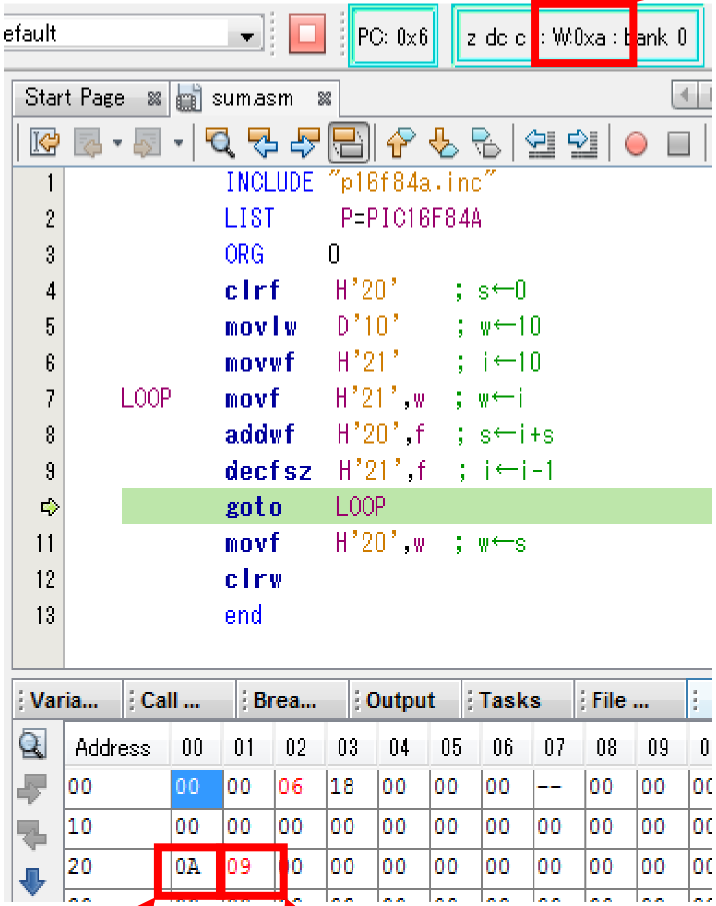


Flow chart



W register

W=10(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
```

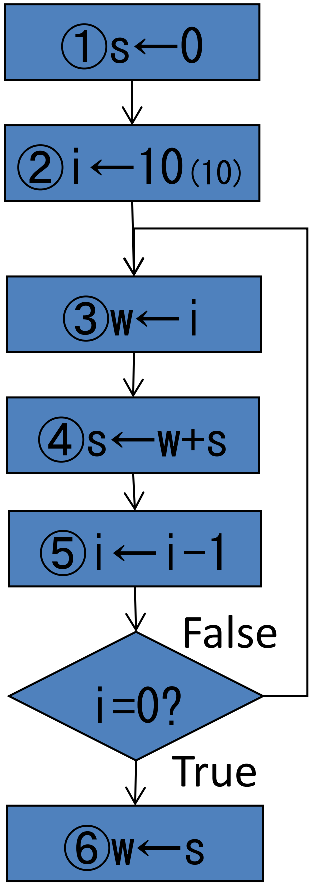
Address	00	01	02	03	04	05	06	07	08	09	0
00	00	00	06	18	00	00	00	--	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00
20	0A	09	00	00	00	00	00	00	00	00	00

s=10(10)

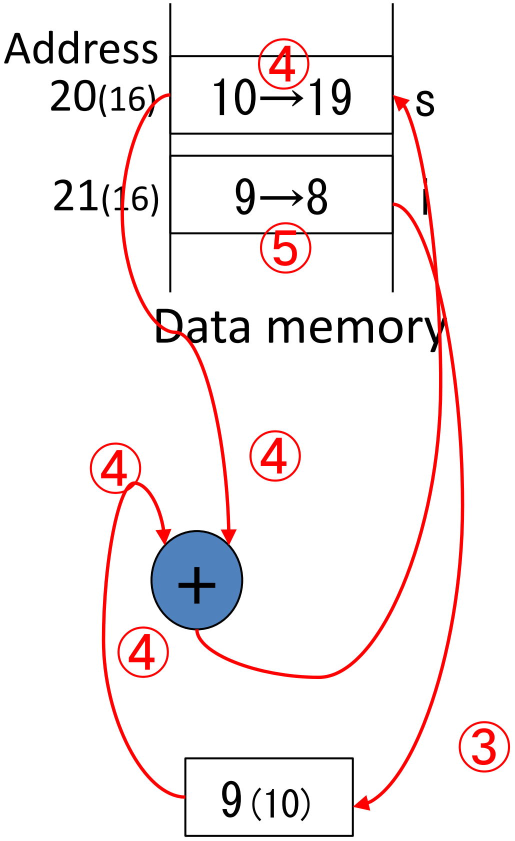
i=9(10)

Simulation Result (3)

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

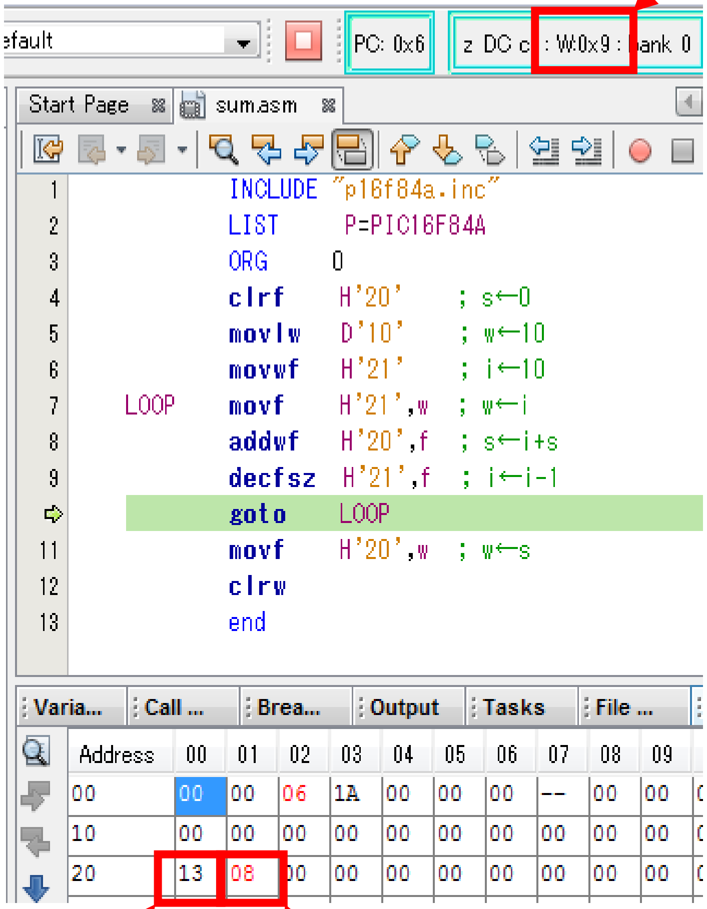


Flow chart



W register

W=9 (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
```

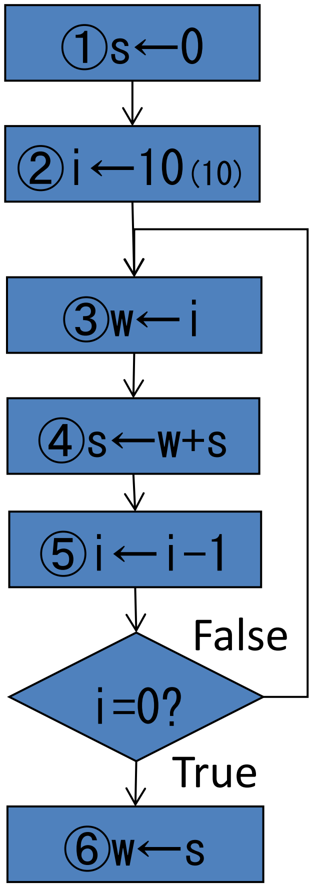
Address	00	01	02	03	04	05	06	07	08	09
00	00	00	06	1A	00	00	00	--	00	00
10	00	00	00	00	00	00	00	00	00	00
20	13	08	00	00	00	00	00	00	00	00

s=19(10)

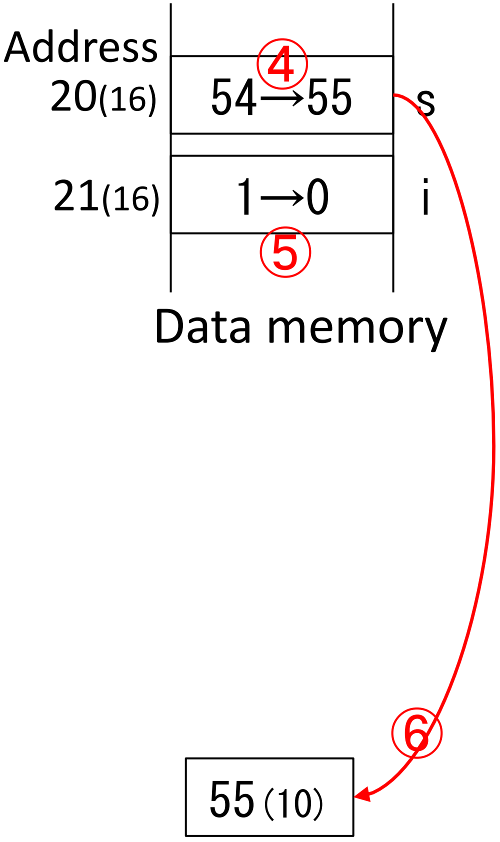
i=8(10)

Simulation Result (4)

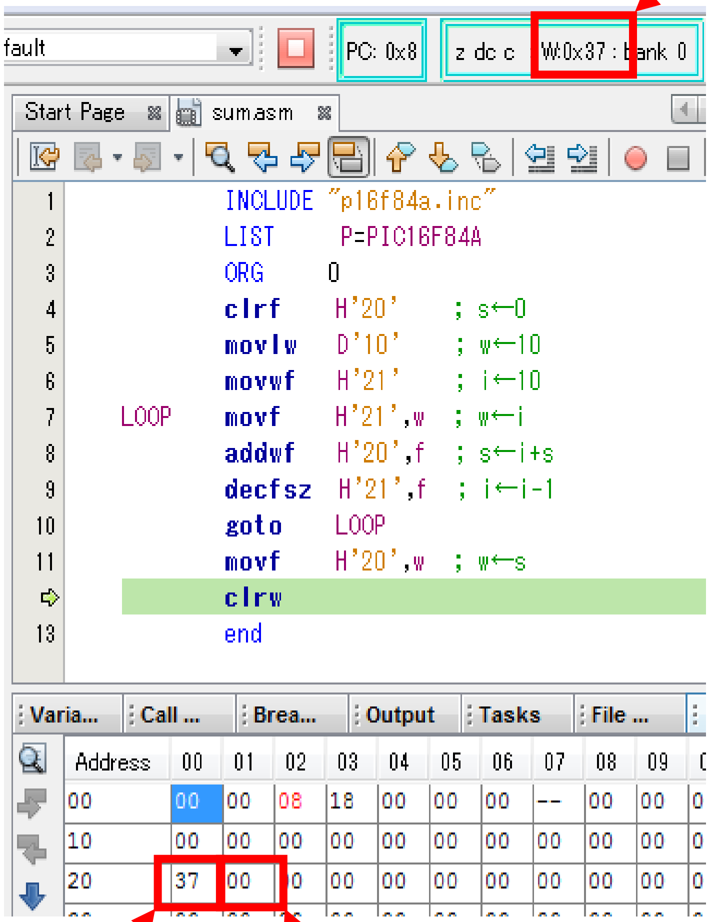
- Simulator: MPLAB X IDE by Microchip
  - The tenth iteration after executed ④⑤⑥ instructions



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  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
```

W=55(10)

s=55(10)

i=0(10)

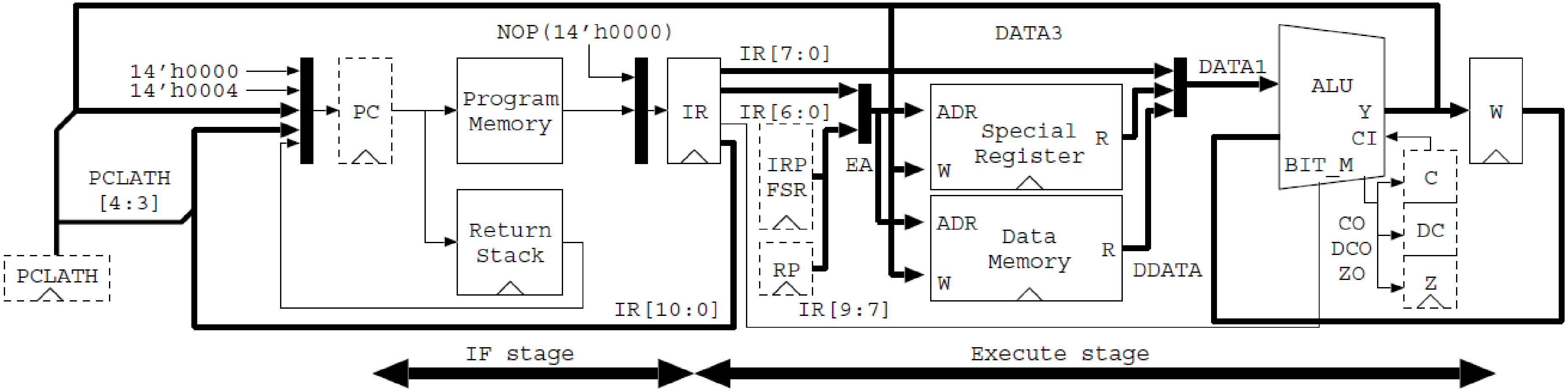
Differences from the original

	Original PIC16	Our design
Pipeline	4 cycles / 1 stage	1 cycle / 1 sgate Different I/O timing
Clock	Selectable clock, external or internal OSC	External clock only
Sleep mode	Low power mode	Repeat NOP instruction
Watch Dog timer (WDT)	Available	None
Timer (TMR0)	Available	None
Prescaler	Available	None
Interrupt	Available	None
Flash mem.	Available	None

Block Diagram

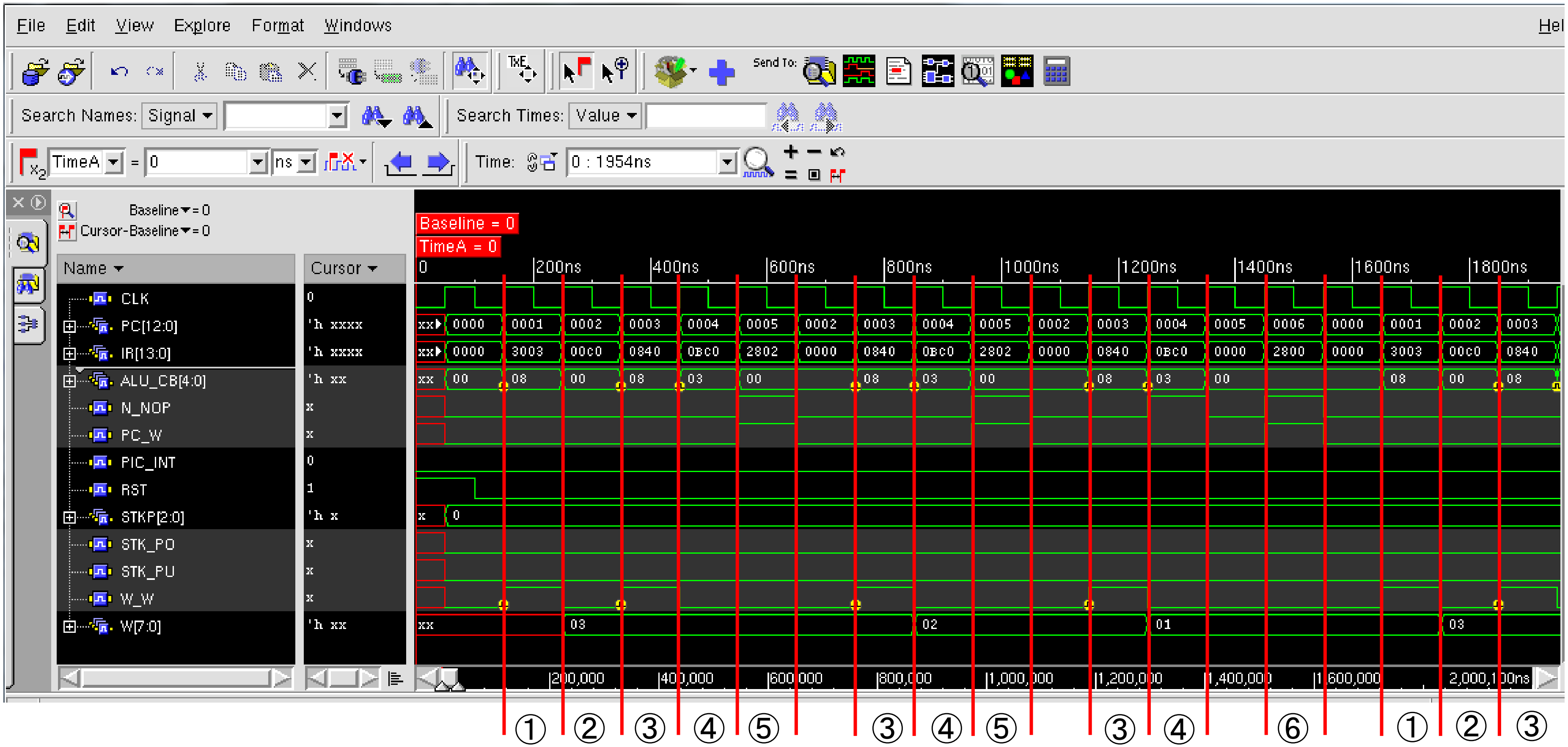
- 2-stage Pipeline
  - Instruction Fetch (IF) stage
  - Execution (E) stage

  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.

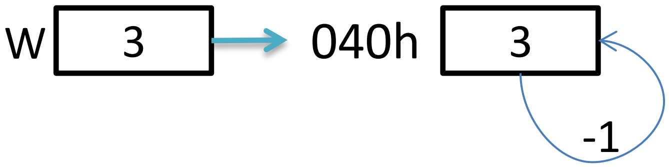


Behavior of 2 Stage Pipeline Processor

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

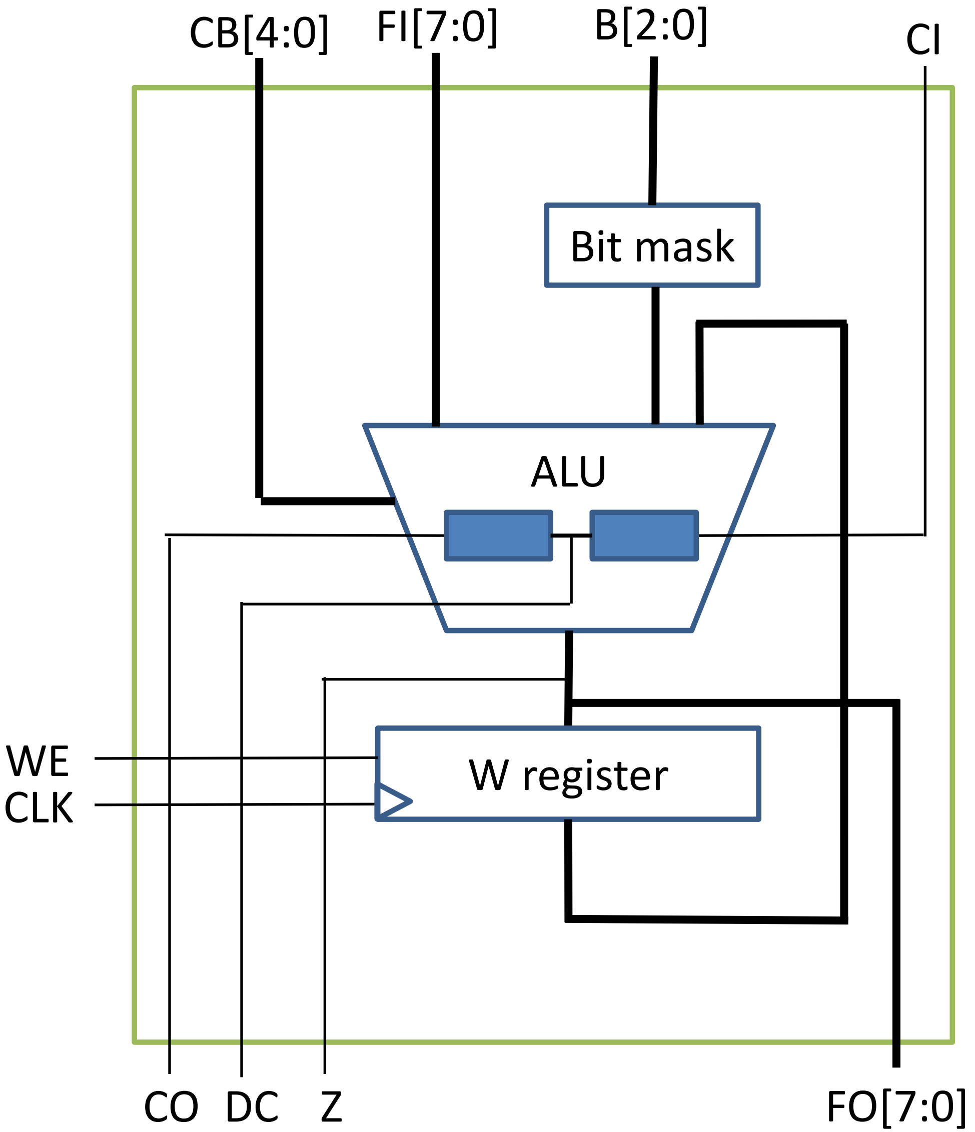
```
0000                00001          ORG  0
0000    3003        00002  L2:   ①MOVLW    3
0001    00C0        00003        ②MOVWF    040h
0002    0840        00004  L1:   ③MOVF     040h, 0
0003    0BC0        00005        ④DECFSZ   040h, 1
0004    2802        00006        ⑤GOTO     L1
0005    2800 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



ALU Design

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	Digit carry for ADD/SUB
Z	Zero flag

alu.v : module alu

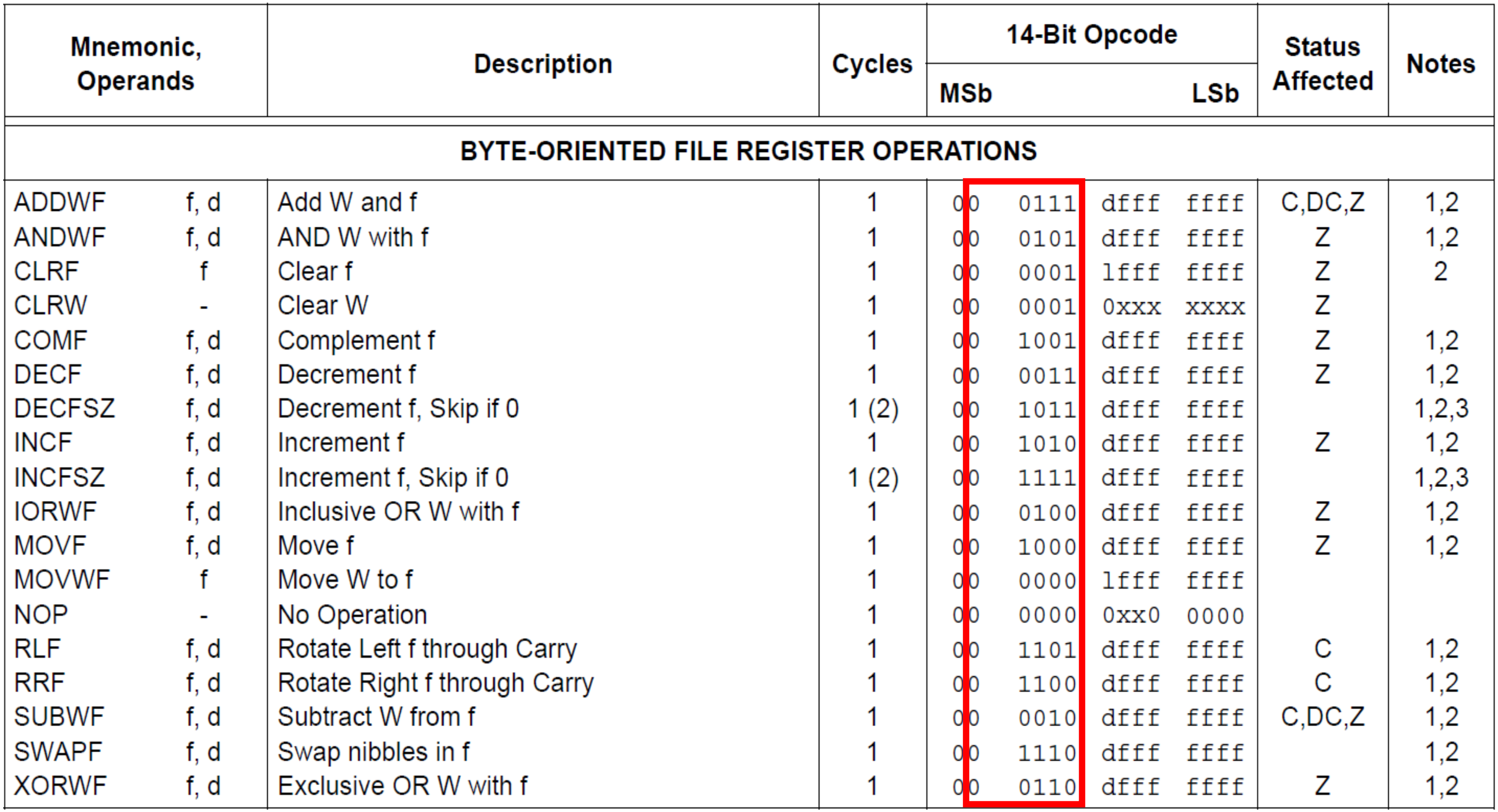
Refer the skeleton code, “alu.v”

```
//
// 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; // Digit Carry for ADD/SUB(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)

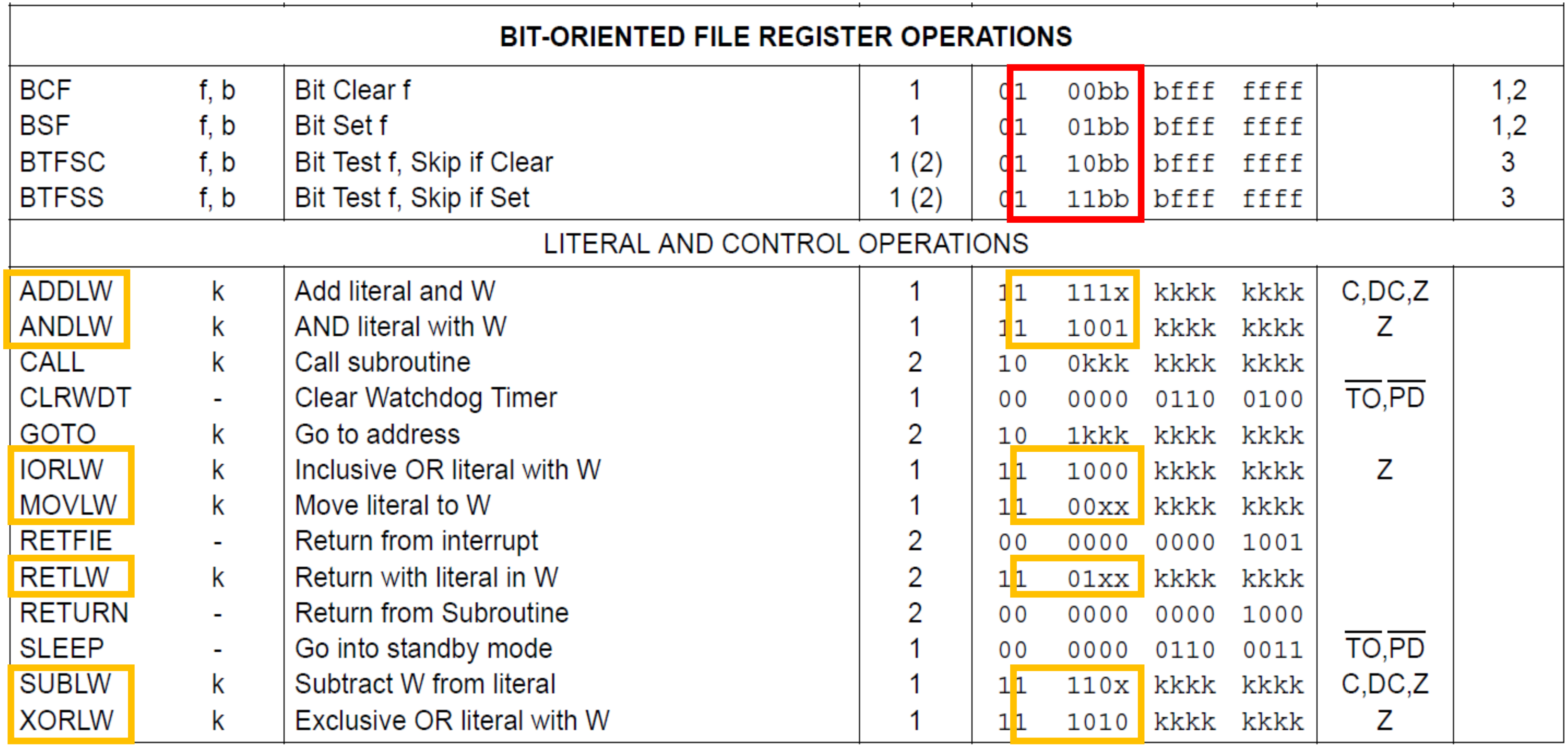


Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	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: Address field of Data memory
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)

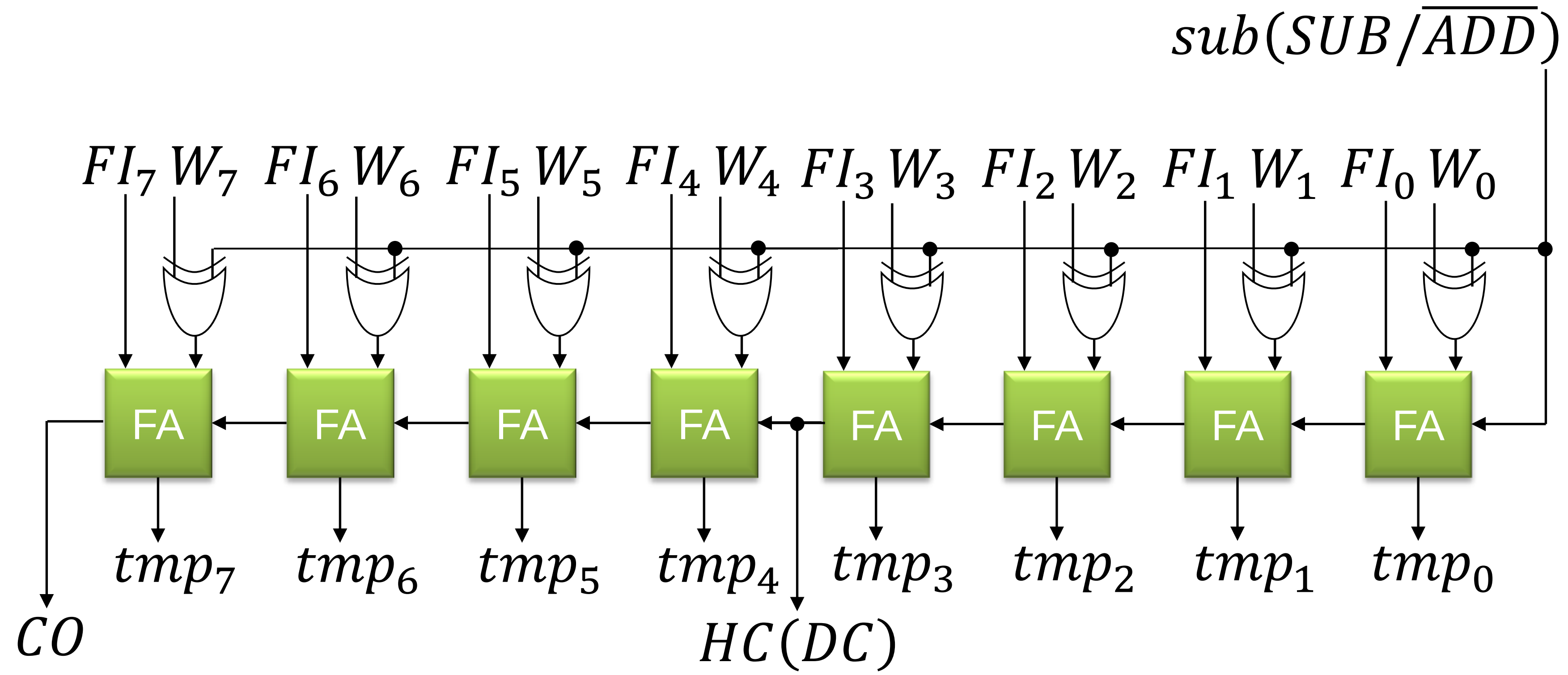


BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
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 field 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
🟨 : ALU codes for LW type inst. are controlled by decoder

ADD/SUB unit for PIC ALU


$sub(SUB/\overline{ADD})$
FI_7 W_7 FI_6 W_6 FI_5 W_5 FI_4 W_4 FI_3 W_3 FI_2 W_2 FI_1 W_1 FI_0 W_0
FA
FA
FA
FA
FA
FA
FA
FA
tmp_7 tmp_6 tmp_5 tmp_4 tmp_3 tmp_2 tmp_1 tmp_0
CO
$HC(DC)$

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

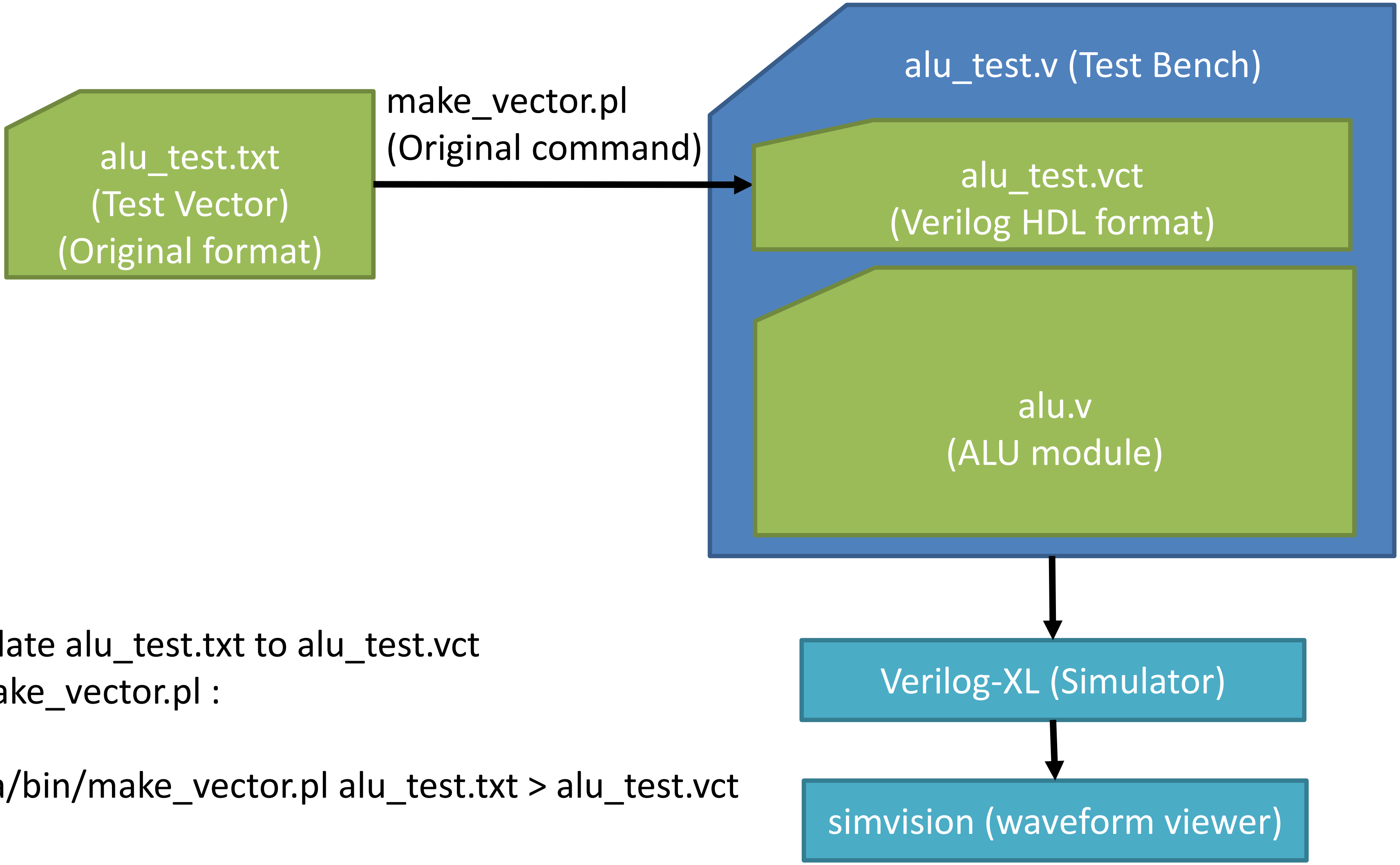
Digit Carry
(ADD:Carry/SUB:$\overline{\text{Borrow}}$)

```
`IADD, `ISUB:
   begin
       { DC, tmp[3:0] } = { 1'b0, ... } + { 1'b0, (...) ? ..., ... } + sub;
             tmp[8:4]   = { 1'b0, ... } + { 1'b0, (...) ? ..., ... } + DC;
```

ALU Simulation

DESIGN EXAMPLE

ALU Simulation



Translate alu_test.txt to alu_test.vct
by make_vector.pl :

```
~kuga/bin/make_vector.pl alu_test.txt > alu_test.vct
```

alu_test.v : Test Bench

```
`timescale 1ns/1ns
```

Time accuracy for simulation (unit time 1ns/precision 1ns)

```
`include "alu_op.v"

module alu_test;
   reg  [7:0] FI;
   reg  [2:0] B;
   reg  [4:0] CB;
   reg        CLK, WE, CO;
```

Input signals (reg type variable)

```
   wire [7:0] FO;
   wire       DC, CI, Z;
```

Output signals (wire type variable)

```
   initial
     begin
        $shm_open("waves.shm");
        $shm_probe("as");
     end
```

Directives for signal monitoring by verilog-XL and simvision, Cadence EDA tools

```
   `include "alu_test.vct"
```

Test vector

```
   alu alu1 ( .CLK(CLK), .CB(CB), .WE(WE), .B(B),
                .FI(FI), .FO(FO), .CI(CO), .CO(CI), .DC(DC), .Z(Z) );

   always @( posedge CLK )
     CO <= CI;
```

Carry flag register

```
endmodule
```

alu_test.txt : test vector

```
# input
CB[4:0]
B[2:0]
FI[7:0]
WE
# clock
CLK 20   # 20ns (50MHz)
# testvector
#  CB[4:0] B[2:0] FI[7:0] WE
10 `IPSF    0      8'hFF    0  # Pass F
20 `IPSF    0      8'h55    0  # Pass F
20 `IPSF    0      8'hAA    0  # Pass F
20 `ICLR    0      8'hAA    0  # Clear F
20 `IINC1   0      8'h00    1  # Increment F, W=1
20 `ISUB    0      8'h01    0  # Subtract 1 from F=1
20 `IDEC1   0      8'h01    1  # Decrement F, W=0
20 `IPSF    0      8'h55    1  # Pass F, W=0x55
20 `IAND    0      8'h55    0  # 0x55 & 0x55
20 `IAND    0      8'hAA    0  # 0xAA & 0x55
20 `IOR     0      8'h55    0  # 0x55 |  0x55
20 `IOR     0      8'hAA    0  # 0xAA | 0x55
20 `IXOR    0      8'h55    0  # 0x55 ^ 0x55
20 `IXOR    0      8'hAA    0  # 0xAA ^ 0x55
20 `IPSW    0      8'h00    0  # Pass W, FO<=W
20 `IRRF    0      8'h55    1  # Rotate Right 0x55, W<= result
20 `IRLF    0      8'hAA    0  # Rotate Left 0xAA
20 `IBSF    2      8'hAA    0  # Bit Set 2
20 `IBCF    1      8'hAA    0  # Bit Clear 1
20 `IBTF    4      8'hAA    0  # Bit Test 4, Check Z flag
20 `IBTF    5      8'hAA    0  # Bit Test 5, Check Z flag
```

Definition of input signals

“# input” is start line of this definition, not comment

Definition of clock signal

“# clock” is start line of this definition, not comment

Definition of test vector

“#testvector” is start line of this def.

One line is one vector.

Column parameters are

- Delay
- CB
- B
- FI
- WE, and
- # comment.

Verilog simulation

- Simulator and waveform viewer are executed on “calc1.st.cs.kumamoto-u.ac.jp”
- For set up the environment, type follows;

```
ssh –X calc1.st.cs.kumamoto-u.ac.jp
source ~kuga/setup/creative2016
```

- For the simulation, type follows;

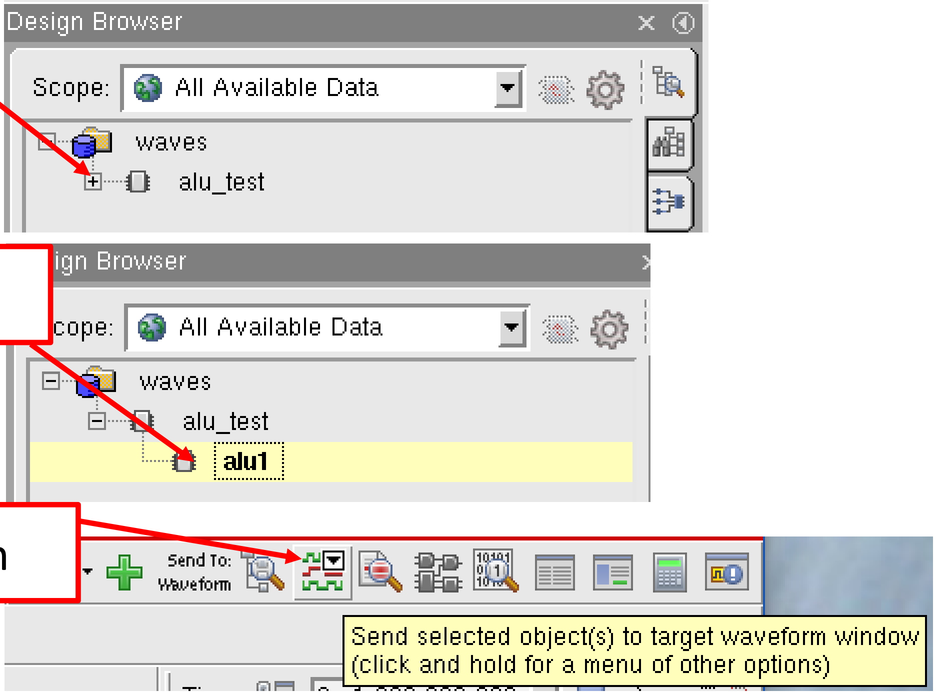
```
verilog alu_test.v alu.v
simvision waves.shm/waves.trn
```

Waveform viewer: simvision

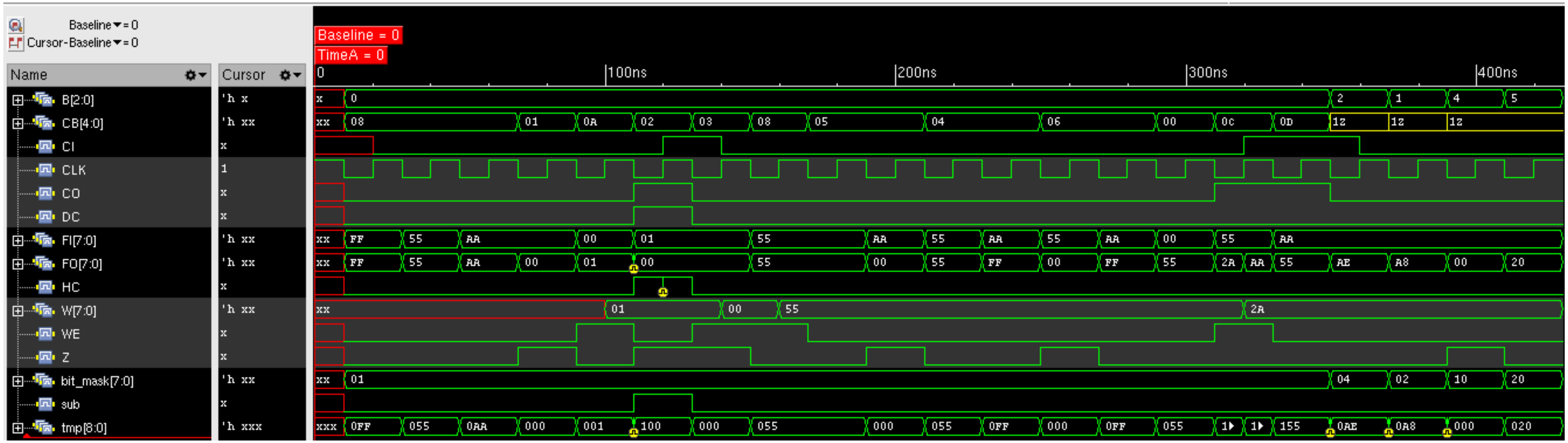
① click “+”

② select “alu1”

② push this icon



Send selected object(s) to target waveform window
(click and hold for a menu of other options)



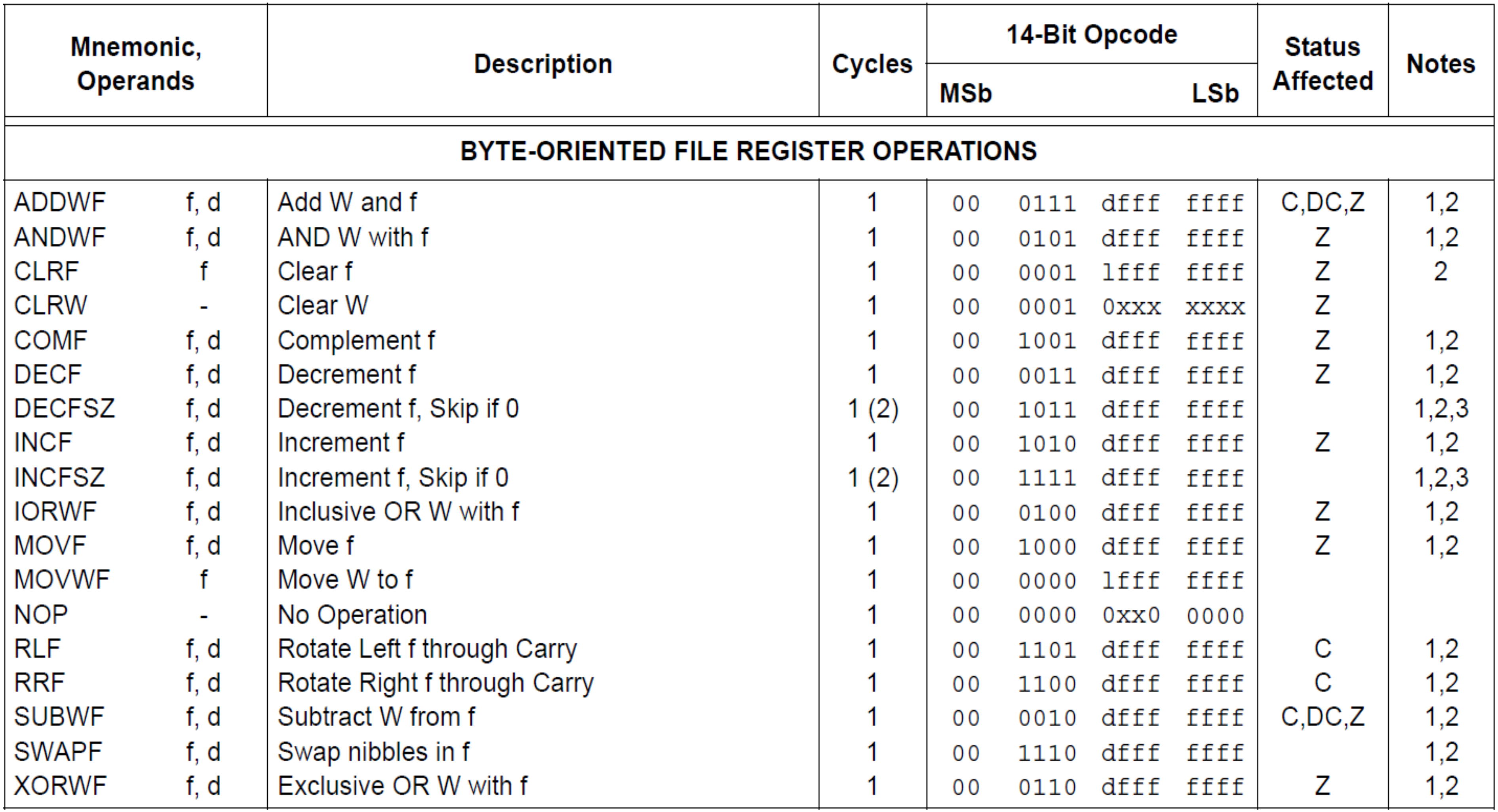
PIC16 core design

DESIGN EXAMPLE

Differences from the original

	Original PIC16	Our design
Pipeline	4 cycles / 1 stage	1 cycle / 1 sgate Different I/O timing
Clock	Selectable clock, external or internal OSC	External clock only
Sleep mode	Low power mode	Repeat NOP instruction
Watch Dog timer (WDT)	Available	None
Timer (TMR0)	Available	None
Prescaler	Available	None
Interrupt	Available	None
Flash mem.	Available	None

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: Address field of Data memory

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

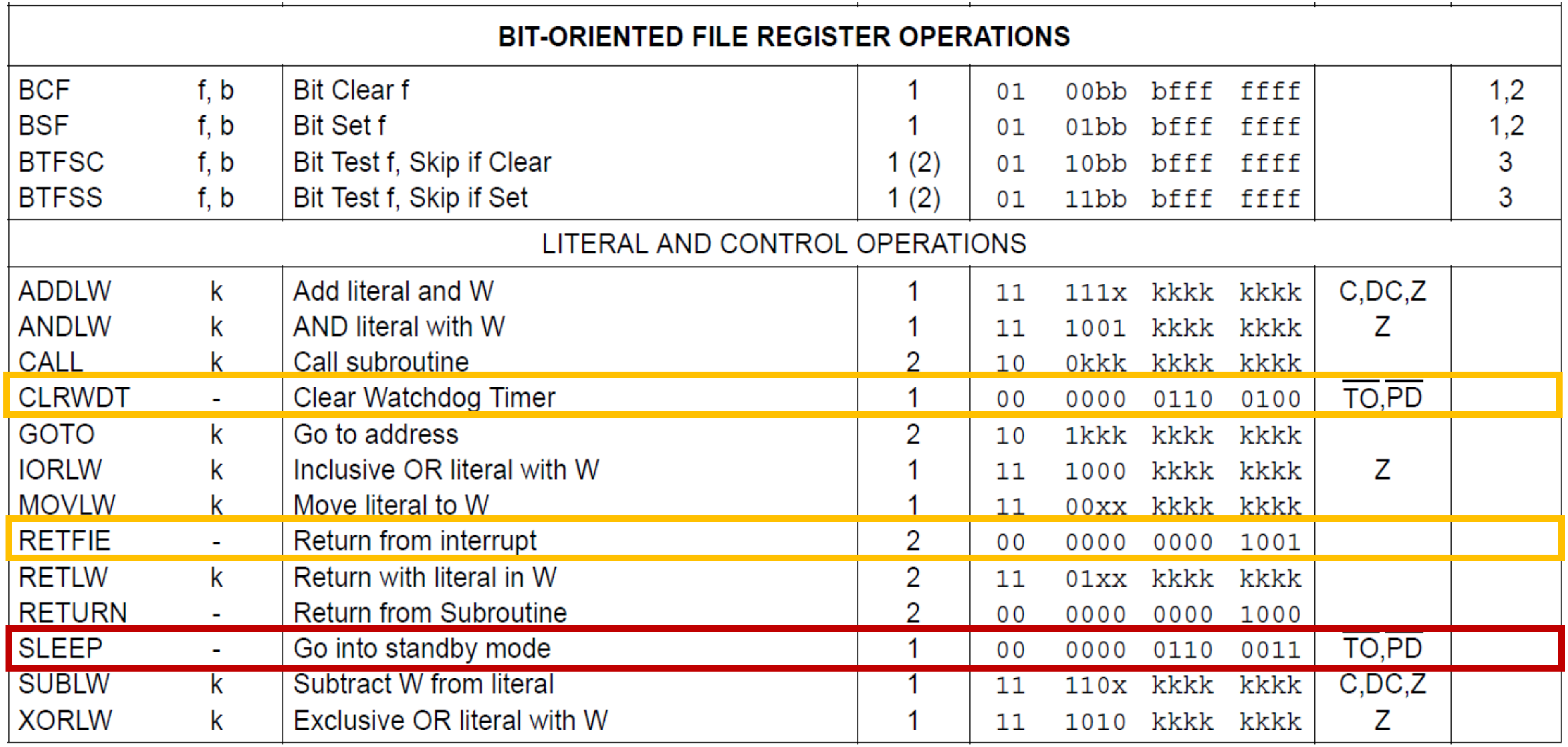
x: Don't care

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

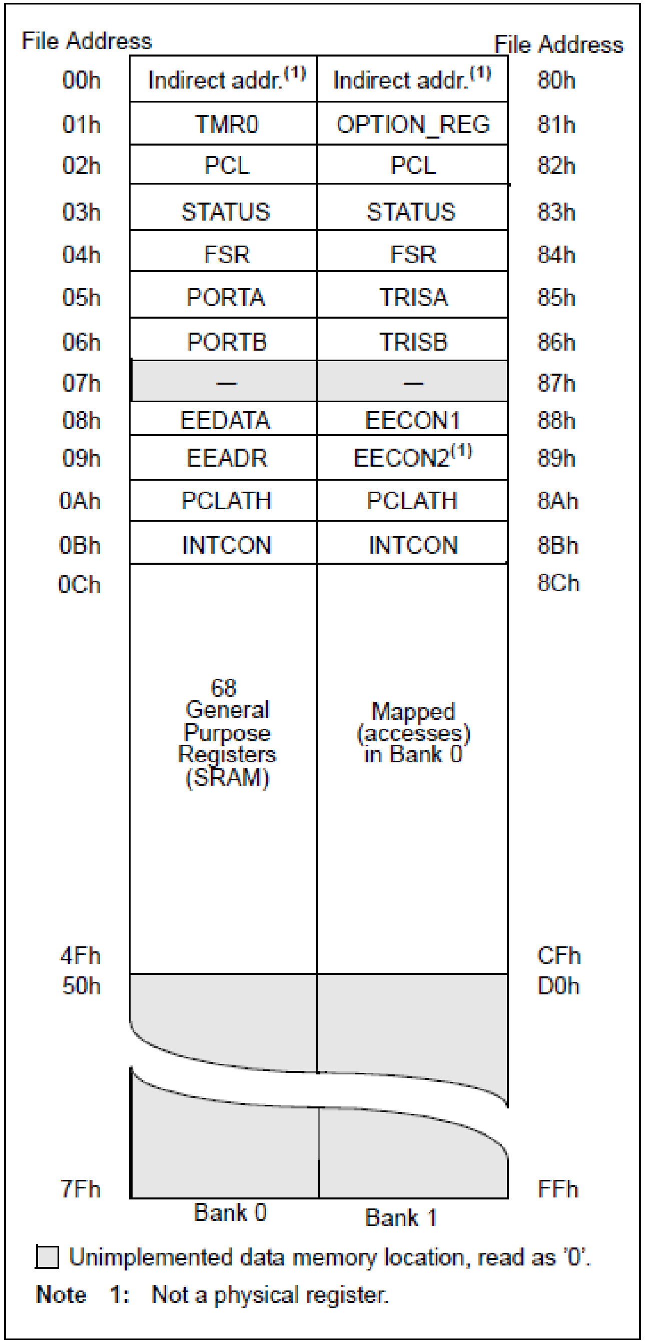
x: Don't care
C: Carry flag, DC: Digit Carry flag
Z : Zero flag
TO, PD: Status for internal states of processor
🟨 : Unimplemented
🟥 : Restricted implementation (cannot wake-up from sleep mode)

Data memory map in our design

RP[1:0]=2’b00		RP[1:0]=2’b01		RP[1:0]=2’b10		RP[1:0]=2’b11	File Address
Indirect addr.(1)	00h	Indirect addr.(1)	80h	Indirect addr.(1)	100h	Indirect addr.(1)	180h
~~TMR0~~	01h	~~OPTION~~	81h	~~TMR0~~	101h	~~OPTION~~	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
	07h		87h		107h		187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
~~INTCON~~	0Bh	~~INTCON~~	8Bh	~~INTCON~~	10Bh	~~INTCON~~	18Bh
Shared RAM	0Ch	Shared RAM	8Ch	Shared RAM	10Ch	Shared RAM	18Ch
	7Fh		FFh		17Fh		1FFh

Data memory map in PIC16F84A

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



File Address			File Address
00h	Indirect addr.[1]	Indirect addr.[1]	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2[1]	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch 4Fh	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch CFh
50h 7Fh	▨	▨	D0h FFh
	Bank 0	Bank 1	

▨ Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

Data memory map in PIC16F648A

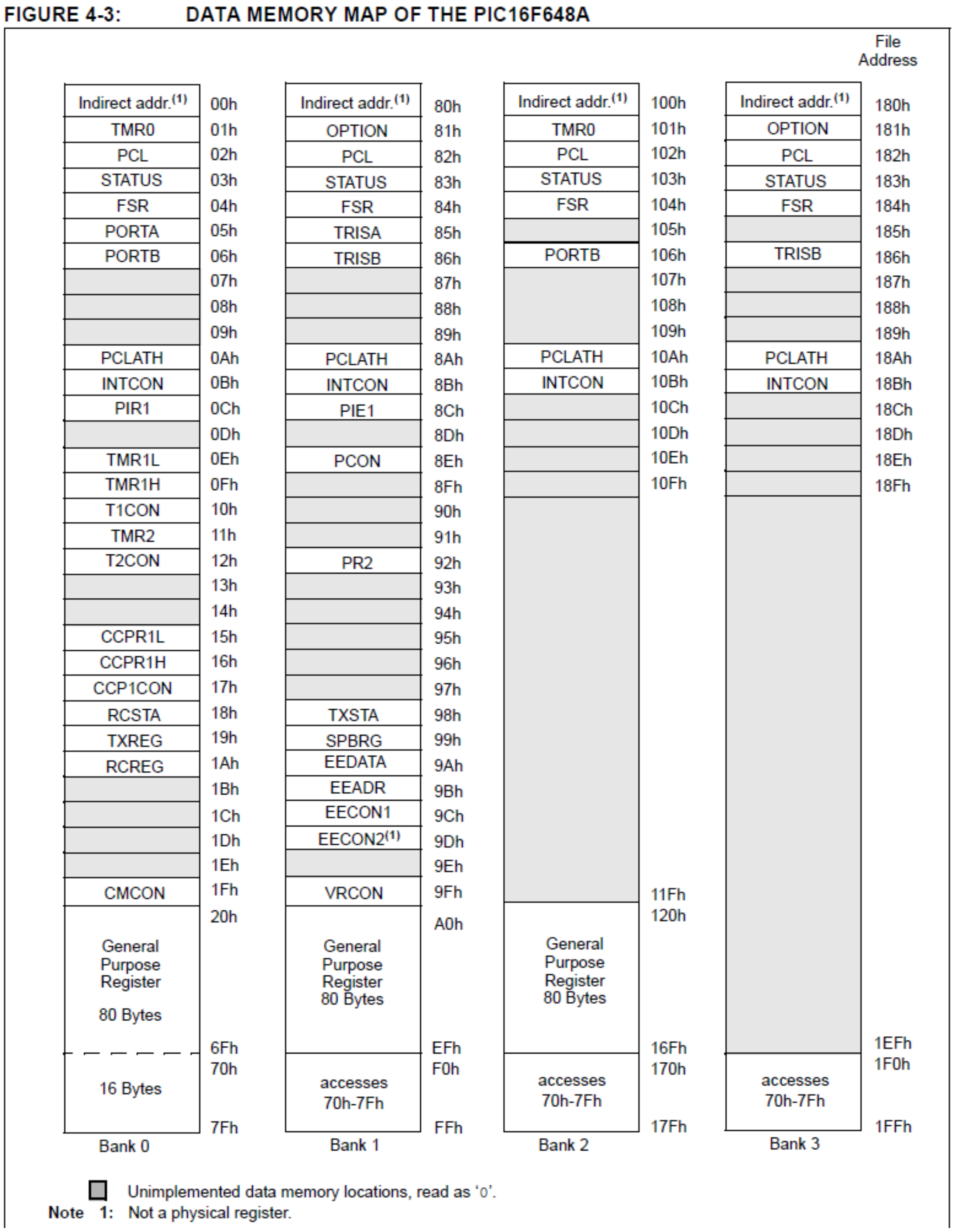


FIGURE 4-3: DATA MEMORY MAP OF THE PIC16F648A

File Address

Indirect addr.[1]	00h	Indirect addr.[1]	80h	Indirect addr.[1]	100h	Indirect addr.[1]	180h
TMR0	01h	OPTION	81h	TMR0	101h	OPTION	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	▨	105h	▨	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
▨	07h	▨	87h	▨	107h	▨	187h
▨	08h	▨	88h		108h	▨	188h
▨	09h	▨	89h		109h	▨	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	▨	10Ch	▨	18Ch
▨	0Dh	▨	8Dh	▨	10Dh	▨	18Dh
TMR1L	0Eh	PCON	8Eh	▨	10Eh	▨	18Eh
TMR1H	0Fh	▨	8Fh	▨	10Fh	▨	18Fh
T1CON	10h	▨	90h	▨		▨	
TMR2	11h	▨	91h				
T2CON	12h	PR2	92h				
▨	13h	▨	93h				
▨	14h	▨	94h				
CCPR1L	15h	▨	95h				
CCPR1H	16h	▨	96h				
CCP1CON	17h	▨	97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah	EEDATA	9Ah				
▨	1Bh	EEADR	9Bh				
▨	1Ch	EECON1	9Ch				
▨	1Dh	EECON2[1]	9Dh				
▨	1Eh	▨	9Eh				
CMCON	1Fh	VRCON	9Fh		11Fh		
General Purpose Register 80 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h		
	6Fh		EFh		16Fh		1EFh
16 Bytes	70h	accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h-7Fh	1F0h
	7Fh		FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

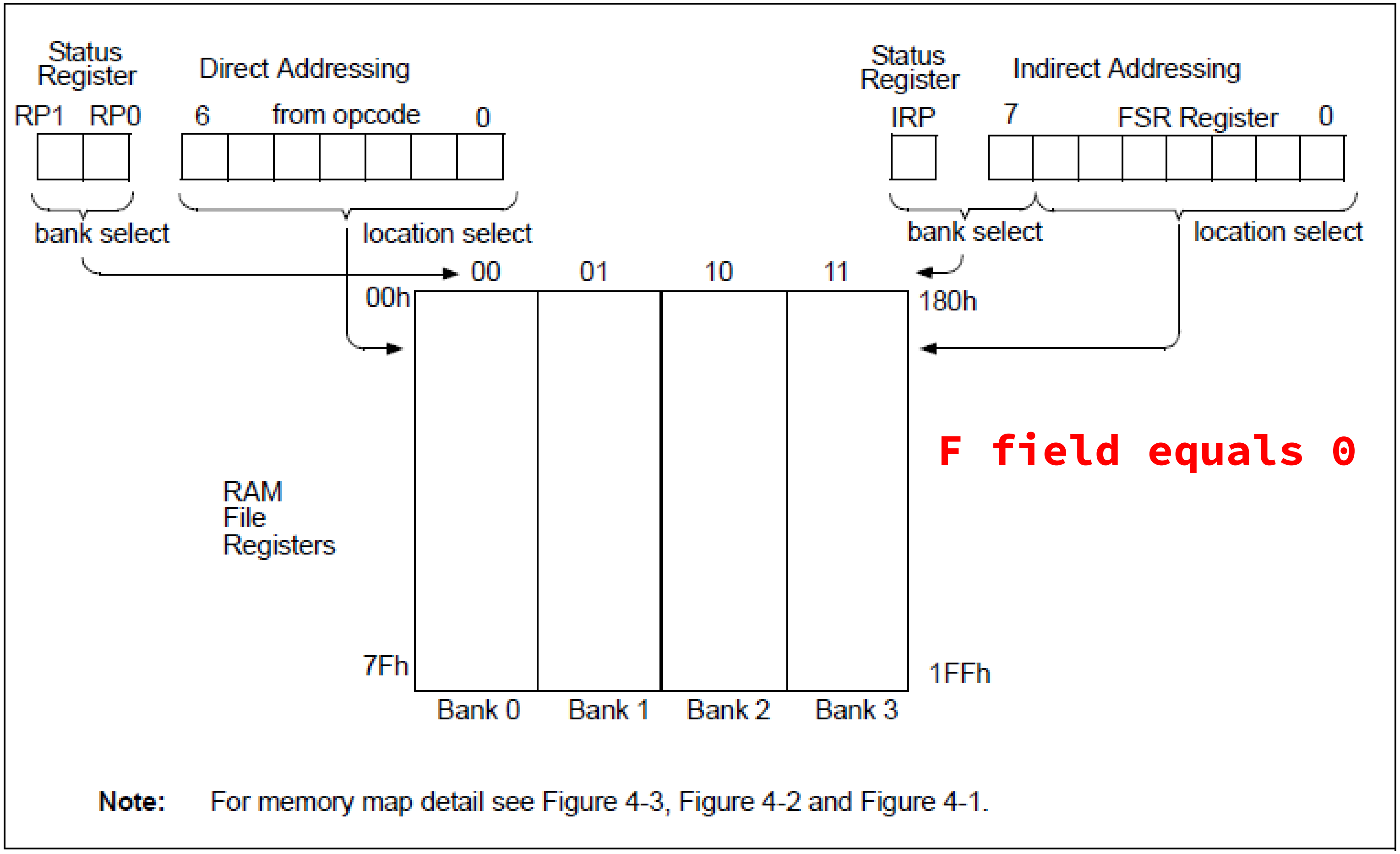
▨ Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

Direct/Indirect Addressing

```
// Effective Addrss
wire [   :   ] EA;
assign EA = ( `IRF ==          ) ? {    ,     } : {    ,     };
```

FIGURE 4-5: **DIRECT/INDIRECT ADDRESSING PIC16F627A/628A/648A**



F field equals 0

Note: For memory map detail see Figure 4-3, Figure 4-2 and Figure 4-1.

Program Counter

```
// Program Counter
  always @( posedge CLK )
    if(              ) PC <=                      ; else // RESET
    if(              ) PC <= {          ,        }; else // CALL, GOTO
    if(              ) PC <= {          ,        }; else // write PCL register
    if(              ) PC <= STK[                ]; else // RETURN, RETLW
    if(              ) PC <=                      ; else // SLEEP mode
                       PC <= PC + 1;

  // Return Stack
  always @( posedge CLK )
    begin
       if(           )       STKP<=       ;                else // for Empty
       if(           ) begin STK[     ]<=   ; STKP<=      ; end else // for CALL
       if(           )                        STKP<=      ;          // for RETxx
    end
```

Return Stack

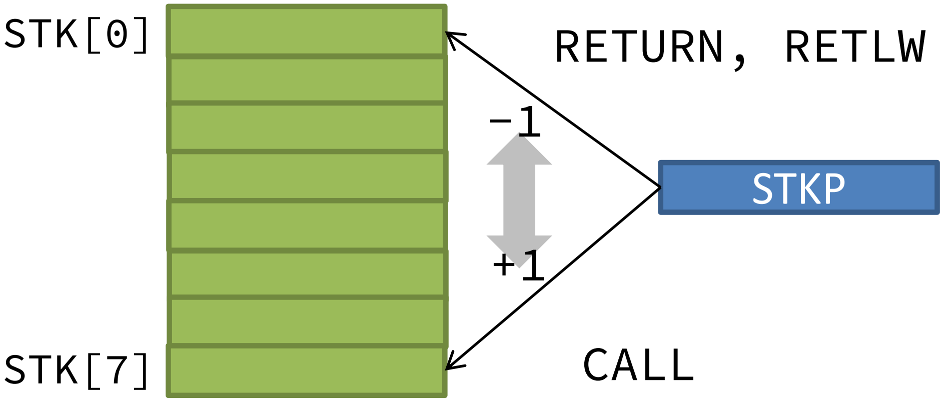
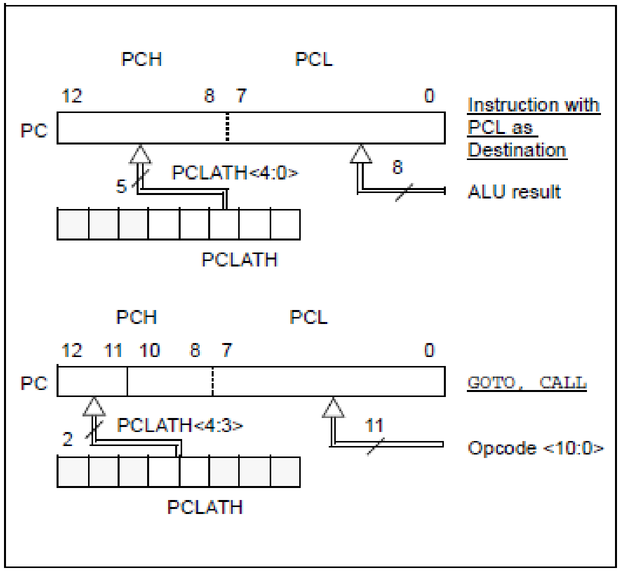
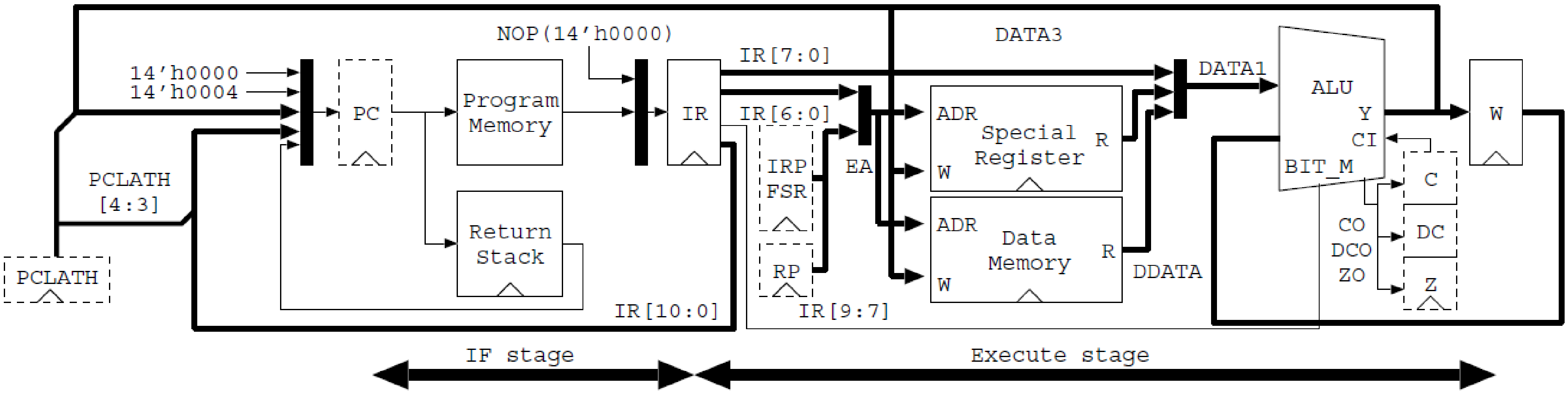


FIGURE 4-4: LOADING OF PC IN DIFFERENT SITUATIONS



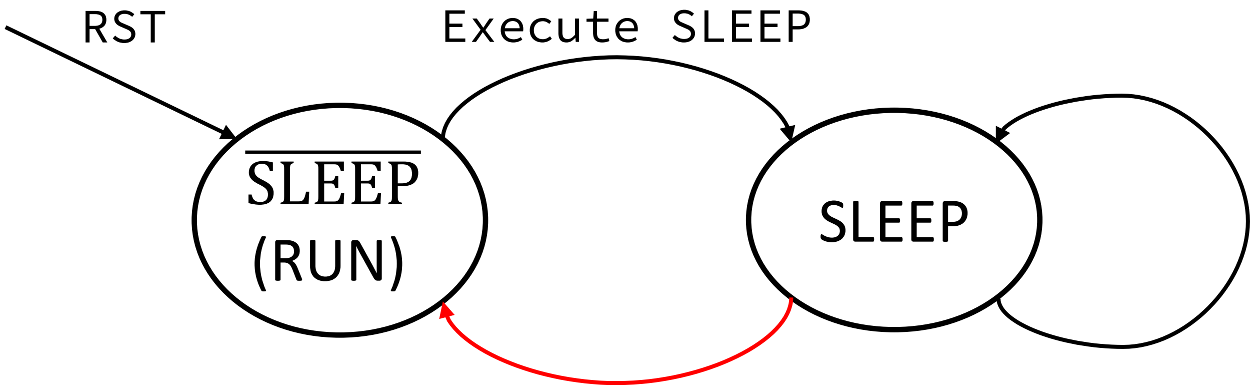
Instruction Memory and Register

```
//
// Instruction Memory (8k word)
//
reg [  :  ] IMEM[  :   ];
initial
  begin
     $readmemh( PROG, IMEM );
  end

//
// Instruction Register
//
always @( posedge CLK )
  if(       ||       ||       ||       )
    IR <=                ; else   // if CALL, RET, cond.SKIP
    IR <=                ;        // Instruction fetch
```



Sleep mode

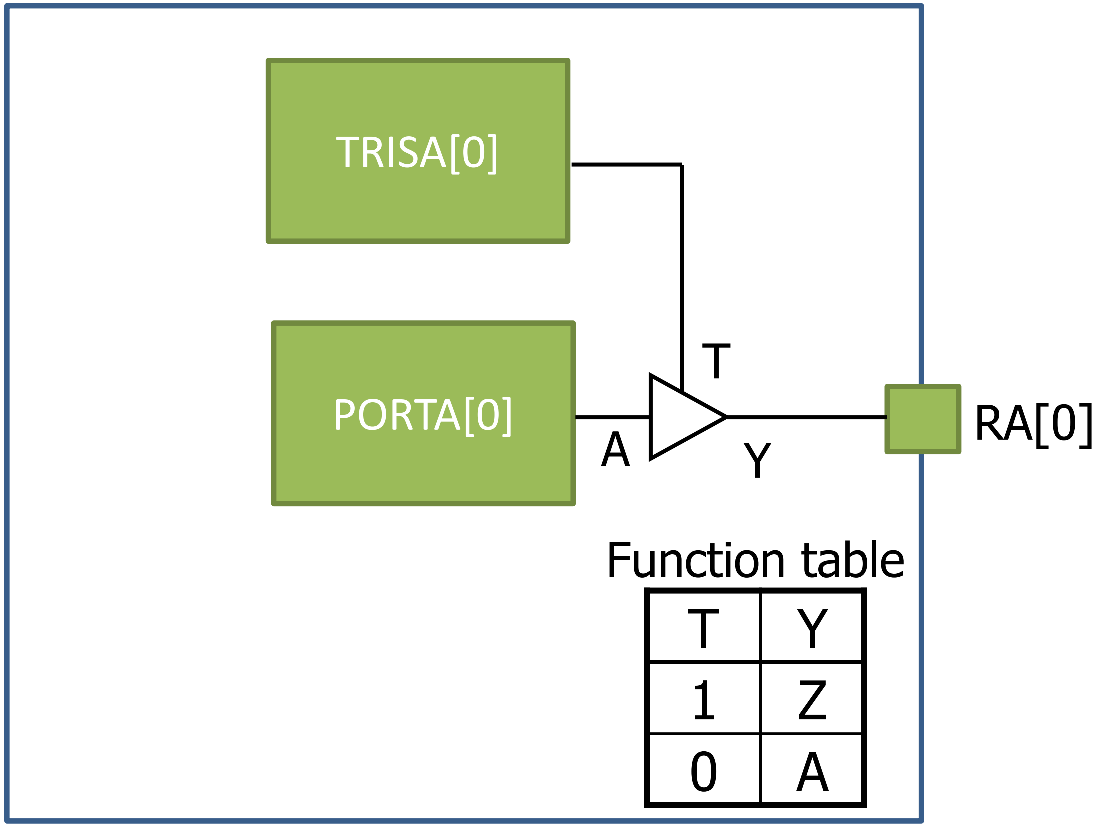
```
// Sleep mode
always @( posedge CLK or posedge RST )
  begin
    if(          ) SLEEP <=   ; else
    if(          ) SLEEP <=   ;
  end
```



Watch Dog Timer overrun
External interrupt
(unimplemented)

Tristate Buffer for GPIO

```
//
// Tristate buffer for GPIO
//
assign RA[0] = ( TRISA[0] ) ? 1'bZ : PORTA[0];
...

assign RB[0] = ( TRISB[0] ) ? 1'bZ : PORTB[0];
...
```



Function table

T	Y
1	Z
0	A

To be continued…