MECE336 Microprocessors I
Addition and Subtraction

Dr. Kurtuluş Erinç Akdoğan

kurtuluserinc@cankaya.edu.tr

Course Webpage: http://MECE336.cankaya.edu.tr

ÇANKAYA ÜNİVERSİTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

Addition: Addition of Two 8bit Numbers

Half Adder Truth Table

Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

→ Without carry input

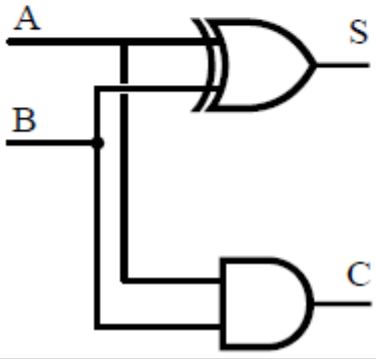
Full Adder Truth Table

Input			Output	
A	B	Cin	S	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

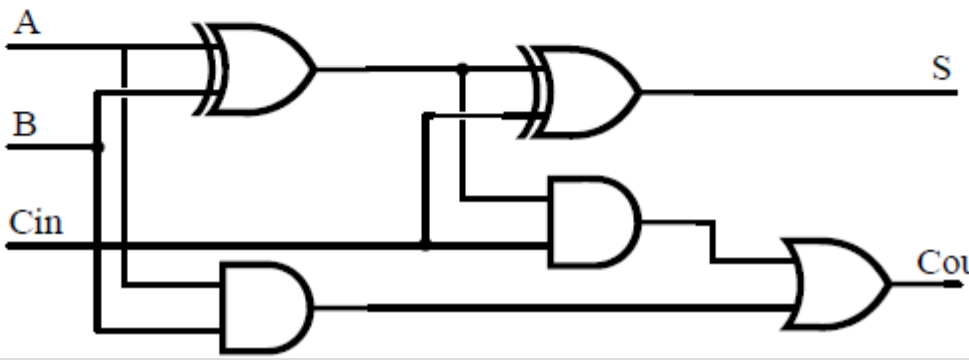
Input			Output	
A	B	Cin	S	Cout
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

→ Addition with carry input

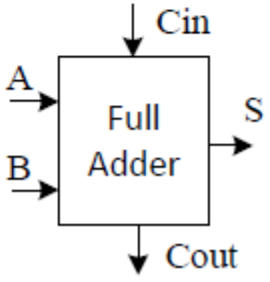
Half Adder



Full Adder



Full Adder Block



Status Register (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0**: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 $\overline{\textbf{TO}}$: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 $\overline{\textbf{PD}}$: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Addition: Instructions **ADDLW**

- addlw k: Add the contents of the working register W and the literal k. Result is in W.
  - Affects C, DC, Z
- Example: The status of the C, DC, and Z flags after the addition of 38H and 2FH in the instructions below:

```
MOVLW 38H  ;move 38H to WREG
ADDLW 2FH  ;add 2FH to WREG and store the result back to WREG
```

```
 38H          0011 1000
+2FH        + 0010 1111
= 67H       = 0110 0111
```

WREG = 67H

- C = 0 there is no carry beyond the D7 bit
- DC = 1 there is a carry from the D3 to the D4 bit
- Z = 0 the WREG has a value other than 0 after the addition

Example

- The status of the C, DC, and Z flags after the addition of 9CH and 64H in the instructions below:

```
MOVLW 9CH   ;move 9CH to WREG
ADDLW 64H   ;add 64H to WREG and store the result back to WREG
```

```
  9CH        1001 1100
+64H       + 0110 0100
=100H      = 0000 0000
WREG = 100H
```

- C = 1 there is a carry beyond the D7 bit
- DC = 1 there is a carry from the D3 to the D4 bit
- Z = 1 the WREG has a value 0 in it after the addition

Example

- The status of the C, DC, and Z flags after the addition of 88H and 93H in the instructions below:

```
MOVLW 88H   ;move 88H to WREG
ADDLW 93H   ;add 93H to WREG and store the result back to WREG
```

```
  88H         1000 1000
+ 93H       + 1001 0011
= 11BH      = 0001 1011
WREG = 11BH
```

- C = 1 there is a carry beyond the D7 bit
- DC = 0 there is no carry from the D3 to the D4 bit
- Z = 0 the WREG has a value other than 0 in it after the addition

http://iamtechnical.com/addlw-instruction-and-status-register

Addition: Instructions **ADDWF**

- addwf f,d: Add the contents of the working register W and the content of file register f. Result is in
  - W if d = 0
  - f of d = 1
- Affects C, DC, Z
- Example: Let's say W=D'15' and the register COUNTER is equal to D'63',initially. Then

  ADDWF COUNTER, 0
  - makes W=15+63=78
- Example:

```
COUNTER EQU 0X20
...
MOVLW 0X05 ; W=0X05
MOVWF COUNTER ; COUNTER=0X05
ADDWF COUNTER, 1 ; COUNTER=COUNTER+W=0X05+0X05=0X0A
```

Addition: Instructions ADDWF Affected Flags, **Z, DC and C**

- **Example:** Let's say W=15 and COUNTER=63, then

  ADDWF COUNTER, F

  - The result is 78 (different from 0) Z=0 at the end of ADDWF instruction.

- **Example:** Let's say W=193 and COUNTER=63, then,

  ADDWF COUNTER,1 ;COUNTER=193+63=256=0

  - The result is zero (256=0 for an 8-bit storage area), and Z=1 at the end of ADDWF instruction.

- **Example:** Let's say W=0X01 and COUNTER=0X0F, then,

  ADDWF COUNTER, 0 ;W=0X01+0X0F=0X10

- The lower nibbles of W and COUNTER are F(=15 in decimal) and 1(=1 in decimal), respectively. The summation of the lower nibbles is 10(=16 in decimal) which means there is a carry from the lower nibble to the upper (F+1=0 with a carry). Therefore at the end of ADDWF operation DC=1.

http://www.onlinepiccompiler.com/InstructionsENG.html#section=ADDWF

Addition: Instructions ADDWF Affected Flags, **Z, DC and C**

- **Example:** Let's say W=0X01 and COUNTER=0X0E, then,

  ADDWF COUNTER, W ;W=0X01+0X0E=0X0F
  - The summation of lower nibbles is F without a carry, then DC=0.

- **Example:** Let's say W=D'50' and COUNTER=D'250', then,

  ADDWF COUNTER, 0 ;W=50+250=300=45
  - The result is 300 and W is an 8- bit register 300-255=45 is hold in W and C=1.

- **Example:** Let's say W=D'190' and COUNTER=D'63', then,

  ADDWF COUNTER, W ;W=190+63=253
  - The result is less than 255 and C becomes 0.

- **Example:** Let's say W=0X0F and COUNTER=0XFF, then,

  ADDWF COUNTER, 0 ;W=0X0F+0XFF=0X0E
  - The result is nonzero, then Z=0. There is an overflow, so C=1. In lower nibbles there is a carry, as well, so DC=1.

http://www.onlinepiccompiler.com/InstructionsENG.html#section=ADDWF

Example

- Add h’4B’ and h’42’. Show the result at PORTB

```
LIST P=16F84A
INCLUDE ‘’P16F84A.INC’’
        CLRF PORTB
        BSF STATUS, 5 ; in BANK1
        CLRF TRISB ;PORTB is output
        BCF STATUS, 5 ; in BANK0
        MOVLW h’4B’ ; W=h’4B’ , b’01001011
        ADDLW h’42’ ; W=h’4B’+ h’42’
        MOVWF PORTB ;PORTB=h’8D’ or b’10001101’ ,C=0
LOOP
        GOTO LOOP
        END
```

Example

- Add 5 to the register 0x22 whenever button at RB0 is pressed. Clear 0x22 if content of 0x22 becomes larger than 255.

```
        list p=16f84a
        include "p16f84a.inc"
        bsf status, 5 ; in bank1
        movlw b'00000001'
        movwf trisb ; RB0 is input
        bcf status, 5 ; in bank0
         clrf portb

loop1 movf portb,0 ; move portb to W register
        andlw b'00000001' ; Check bit 1
        btfsc   status,2   ; test the z flag to see if it is not SET
        goto    loop1   ;z flag is set
        movlw .5 ;if z flag is not set
        addwf 0X22,1 ;add 5 to register 0x22
        btfss status,0; test the c flag to see if it is SET
        goto    loop1   ;c flag is not set
        clrf 0x22; if c flag is set clear the content of register 0x22
        goto    loop1   ;
        end
```

Addition of two 16-bit numbers:

- If the numbers greater than 1 byte (8 bit), we can add these numbers using 16-bit addition. When adding two 16-bit data operands, we need to be concerned with the propagation of a carry from the lower byte to the higher byte. This is called multi-byte addition to distinguish it from the addition of individual byte.
- For example look at the addition of h'3CE7'+h'3B8D'

```
    1
  3C E7
+ 3B 8D
-------
  78 74
```

- When the first byte is added, there is a carry (E7+8D=74, C=1). The carry is propagated to the higher byte, which results in 3C+3B+1=78

Addition of two 16-bit numbers: EXAMPLE

- Write a program to add two 16-bit numbers. The numbers are h'3CE7' and h' 3B8D'. Show low byte of the result at PORTB. When bit_1 of PORTA (RA1) is pressed, show high byte of the result at PORTB.

Solution: 2 byte (16-bit) numbers;

- A=3CE7, B= 3B8D
- Low byte of A (AL)=E7, High byte of A (AH)=3C
- Low byte of B (BL)=8D, High byte of B (BH)=3B

```
;=======16-bit addition =======
    LIST        P=16F84A
    INCLUDE   ''P16F84A.INC''
              CLRF     PORTB
              BSF      STATUS, 5    ; in BANK1
              CLRF     TRISB         ;PORTB is output
              MOVLW  h'FF'
              MOVWF  TRISA        ; PORTA is input
              BCF       STATUS, 5    ; in BANK0
        AL      EQU      h'0C'       ; Address of AL
        AH      EQU      h'0D'       ; Address of AH
        BL      EQU      h'0E'       ; Address of BL
        BH      EQU      h'0F'       ; Address of BH
BEGIN
              MOVLW  h'E7'         ; W=h'E7'
              MOVWF  AL           ;AL=h'A3'
              MOVLW  h'3C'         ; W=h'3C'
              MOVWF  AH           ;AH=h'3C'
              MOVLW  h'8D'         ; W=h'8D'
              MOVWF  BL           ;BL=h'8D'
              MOVLW  h'3B'         ; W=h'3B'
              MOVWF  BH           ;BH=h'3B'
```

```
;====cont. Prog====
ADD
        MOVF    AL,W            ;W=AL
        ADDWF   BL,F            ;BL=BL+W(AL)
        BTFSC   STATUS, 0       ;C=0 or 1?
        INCF    BH,F            ;if C=1, BH=BH+1
        MOVF    AH,W            ;W=AH
        ADDWF   BH,F            ;BH=BH+W(AH)
SHOW_LOW_BYTE
        MOVF    BL,W            ;W=BL
        MOVWF   PORTB           ;show low byte at PORTB
TEST_RA1
        BTFSC   PORTA,1         ;RA1 is press
        GOTO    TEST_RA1        ;if NO
SHOW_HIGH_BYTE
        MOVF    BH,W            ; if YES , W=BH
        MOVWF   PORTB           ;show high byte at PORTB
LOOP
        GOTO    LOOP
        END
```

Subtraction: Background Twos-Complement

- Binary operation that can be used for subtraction
- Computation for a given binary number B
  - Take the bitwise complement of B (called ones-complement)
  - Add 1 to the result
- Examples: suppose we want to find how -28
- First we write out 28 in binary form.

  00011100
- Then we invert the digits. 0 becomes 1, 1 becomes 0.

  11100011
- Then we add 1.

  11100100
- That is how one would write -28 in 8 bit binary.

Subtraction: Background

Subtraction of Two Binary Numbers: B1 - B2

- Compute the twos-complement of B2
  - Add B1 and the twos-complement of B2
  - Result is B1 - B2
  - If the result is negative, there is "borrow" indicated with C flag is zero
- Examples

```
 (+8) 0000 1000                0000 1000
-(+5) 0000 0101 -> Negate -> +1111 1011
-----                         -----------
 (+3)                         1 0000 0011 : discard carry-out
```

```
 (+3)   0000 0011
+(-8)   1111 1000
-----------------
 (-5)   1111 1011
```

Subtraction: Instructions SUBWF

- Substract Working Register from File Register
- subwf f,d: Subtract the W register from the content of memory location f. Result is written in
  - Working register W if d = 0
  - File register f if d = 1
- The C/borrow flag (bit 0) in the Status register is
  - 0 if there is borrow
  - 1 if there is no borrow

Example

- Write a program to subtract h’52’ - h’53’. Show the result at PORTB.

```
;======8_bit subtraction====
    LIST        P=16F84A
    INCLUDE   ‘’P16F84A.INC’’
                CLRF      PORTB
                BSF       STATUS, 5     ; in BANK1
                CLRF      TRISB         ;PORTB is output
                BCF       STATUS, 5     ; in BANK0
                MOVLW  h’52’            ; W=h’52’
                MOVWF   PORTB           ;PORTB=52
                MOVLW    h’53’          ; W=h’53’
                SUBWF    PORTB,F        ;PORTB=PORTB (h’52’ )-W(h’53’),result
negative
                COMPF     PORTB
                INCF        PORTB       ;2’s complement os result,
 LOOP
            GOTO    LOOP
            END
```

Subtraction: Instructions SUBLW

- Substract Working Register from Literal
- sublw k: Subtract the W register from a literal k. Result is written into W.
- The C/borrow flag (bit 0) in the Status register is
  - 0 if there is borrow
  - 1 if there is no borrow

Examples

- Example

```
movlw 0
sublw 0
```

- Means load W with 0x00. Subtract that from 0x00.
- Subtraction is by complementing the W register and adding 1 (2's complement), and adding to the literal.
- 0-0= 0xFF + 1 + 0x00 = 0x00 (C set)

- Example
- In general, the C bit (really a borrow rather than carry for subtraction) is set when the result is positive (including zero), as is normal in 2's complement subtraction.

```
movlw 0x00
sublw 0x33
```

- 0x33-0x00= 0xFF + 1 + 0x33 = 0x33 (C set)

Example: Addition of two 16-bit numbers

```
 list p=16f84a
   include "p16f84a.inc"
   __config _CP_OFF&_WDT_OFF&_XT_OSC

AL  EQU       0x0C;
AH  EQU       0x0D;
BL  EQU       0x0E;
BH  EQU       0x0F;
RL  EQU       0x10;
RH  EQU       0x11;
CH  EQU       0x12;

   org 0
main
   movlw   ;
   movwf   AL;
   movlw   ;
   movwf   AH;            A = AH AL = 0xF4A3 = 1111 0100 1010 0011 = 62627

   movlw   ;
   movwf   BL;
   movlw   ;
   movwf   BH;            B = BH BL = 0x146E = 0001 0100 0110 1110 = 5230

   call    add_16; call 16-bit adder subroutine
   goto    end_label;
```

Example Continued: Addition of two 16-bit numbers

```
add_16;     16-bit adder subroutine
   movf    AL,0;
   addwf   BL,0;
   movwf   RL;  RL = AL + BL;

   movf    STATUS,0; move status register to W
   andlw   b'000000001'; W = 00000000 if C flag = 0, W = 00000001 if C flag = 1

   addwf   AH,0; W = AH + 0 if C flag = 0, W = AH + 1 if C flag = 1
   movwf   RH; RH = AH if C flag = 0, W = AH + 1 if C flag = 1

   movf    STATUS,0;
   andlw   b'00000001'; W = 00000000 if C flag = 0, W = 00000001 if C flag = 1
   movwf   CH;            save C flag in CH

   movf    BH,0;
   addwf   RH,1;   result RH = RH + BH

   btfsc   STATUS,0;
   goto    return_label; C flag is already 1
   btfsc   CH,0;
   bsf     STATUS,0;W = 00000000 if C flag = 0, W = 00000001 if C flag = 1
return_label;
   return;

end_label;
   end;
```