Lecture 4

9/14/16

The instruction

comments begin with semicolon

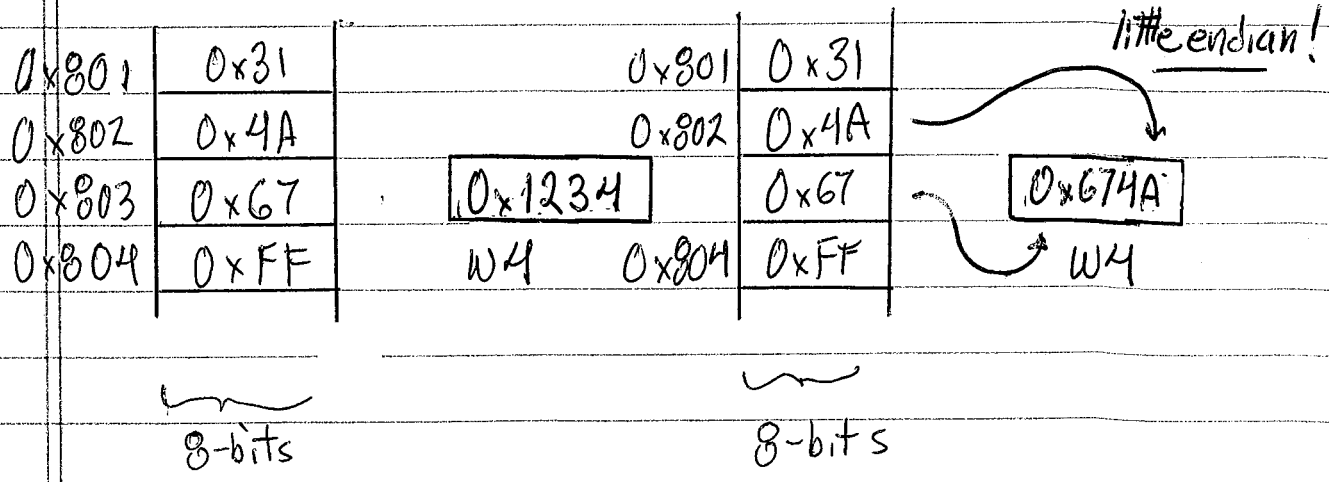
```
mov W3, W5    ; copy value in W3 to W5
```

is an example of register direct addressing.

File register addressing is used to move a value from a register to data memory or from data memory to a register. This is done by specifying a location in memory in the mov instruction.

Example: Copy the value in data memory at location 0x0802 into W4

```
mov 0x0802, W4
```



MOV

Move f to Wnd

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
	X	X	X	X	X	X

Syntax: {label:} MOV f, Wnd

Operands: f ∈ [0 ... 65534]
Wnd ∈ [W0 ... W15]

Operation: (f) →Wnd

Status Affected: None

Encoding:

1000	0fff	ffff	ffff	ffff	dddd

Description: Move the word contents of the specified file register to Wnd. The file register may reside anywhere in the 32K words of data memory, but must be word-aligned. Register direct addressing must be used for Wnd.

The 'f' bits select the address of the file register.
The 'd' bits select the destination register.

Note 1: This instruction operates on word operands only.

2: Since the file register address must be word-aligned, only the upper 15 bits of the file register address are encoded (bit 0 is assumed to be '0').

3: To move a byte of data from file register memory, the "`MOV f to Destination`" instruction (page 279) may be used.

Words: 1

Cycles: 1(1)

Note 1: In dsPIC33E and PIC24E devices, the listed cycle count does not apply to read and read-modify-write operations on non-CPU Special Function Registers. For more details, see **Note 3** in **Section 3.2.1 "Multi-Cycle Instructions"**.

Example 1:

```
MOV   CORCON, W12     ; move CORCON to W12
```

	Before Instruction	After Instruction
W12	78FA	00F0
CORCON	00F0	00F0
SR	0000	0000

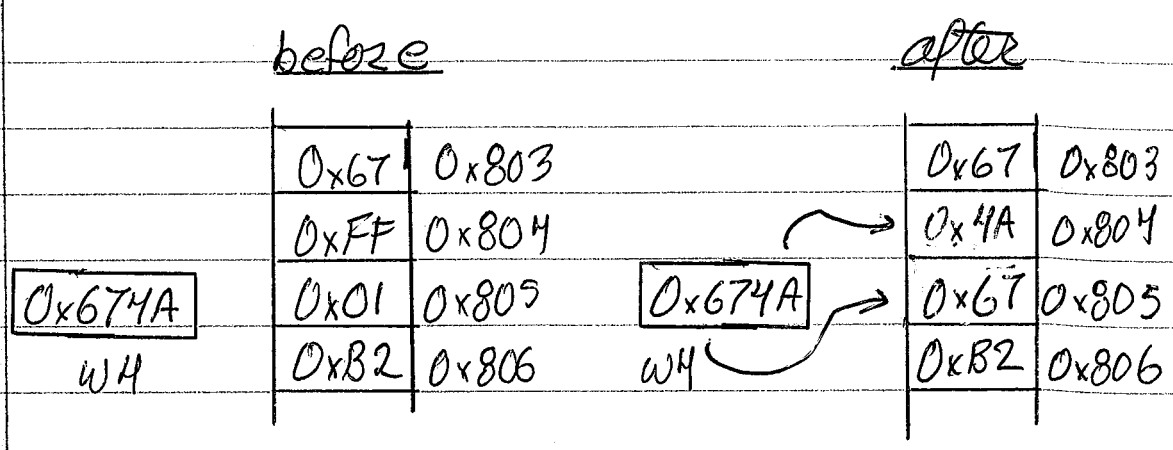
Example 2:

```
MOV   0x27FE, W3    ; move (0x27FE) to W3
```

	Before Instruction	After Instruction
W3	0035	ABCD
Data 27FE	ABCD	ABCD
SR	0000	0000

File register addressing can also be used to copy

mov W4, 0x0804



There are some restrictions, moving data between a file register and a working register can only be done with 16-bit words. Both the source and the destination cannot be file registers

Note that Microchip refers to data memory locations as file registers. Most other microcontroller vendors would use the term memory direct addressing to indicate the memory address is directly specified. The addresses specified with this mov instruction are absolute addresses.

MOV

Move Wns to f

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
	X	X	X	X	X	X

Syntax: {label:} MOV Wns, f

Operands: f ∈ [0 ... 65534]
Wns ∈ [W0 ... W15]

Operation: (Wns) →f

Status Affected: None

Encoding:

1000	1fff	ffff	ffff	ffff	ssss

Description: Move the word contents of the working register Wns to the specified file register. The file register may reside anywhere in the 32K words of data memory, but must be word-aligned. Register direct addressing must be used for Wn.

The 'f' bits select the address of the file register.
The 's' bits select the source register.

Note 1: This instruction operates on word operands only.

2: Since the file register address must be word-aligned, only the upper 15 bits of the file register address are encoded (bit 0 is assumed to be '0').

3: To move a byte of data to file register memory, the "`MOV WREG to f`" instruction (page 280) may be used.

Words: 1

Cycles: 1

Example 1:

```
MOV   W4, XMDOSRT     ; move W4 to XMODSRT
```

	Before Instruction		After Instruction
W4	1200	W4	1200
XMODSRT	1340	XMODSRT	1200
SR	0000	SR	0000

Example 2:

```
MOV    W8, 0x1222     ; move W8 to data address 0x1222
```

	Before Instruction		After Instruction
W8	F200	W8	F200
Data 1222	FD88	Data 1222	F200
SR	0000	SR	0000

Literal Addressing

The move 16-bit literal to register instruction use literal addressing, i.e. the value, instead of the address of the value, is in the instruction.

Example: load 0x8765 into register W3

```
mov  #0x8765, W3
```

	before	after
W3	0x5678	0x8765
W4	0x9ABC	0x9ABC

This would be encoded as

0x8765 (W3)

0010 1000 0111 0110 0101 0011

or

0x287653

Note that the literal can be signed or unsigned

```
mov #-30875, W3
```

MOV

Move 16-bit Literal to Wnd

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
	X	X	X	X	X	X

Syntax: {label:} MOV #lit16, Wnd

Operands: lit16 ∈ [-32768 ... 65535]
Wnd ∈ [W0 ... W15]

Operation: lit16 →Wnd

Status Affected: None

Encoding:

0010	kkkk	kkkk	kkkk	kkkk	dddd

Description: The 16-bit literal 'k' is loaded into Wnd. Register direct addressing must be used for Wnd.

The 'k' bits specify the value of the literal.
The 'd' bits select the address of the working register.

Note 1: This instruction operates only in Word mode.
2: The literal may be specified as a signed value [-32768:32767], or unsigned value [0:65535].

Words: 1

Cycles: 1

Example 1:

```
MOV   #0x4231, W13     ; load W13 with #0x4231
```

	Before Instruction	After Instruction
W13	091B	4231
SR	0000	0000

Example 2:

```
MOV   #0x4, W2         ; load W2 with #0x4
```

	Before Instruction	After Instruction
W2	B004	0004
SR	0000	0000

Example 3:

```
MOV   #-1000, W8       ; load W8 with #-1000
```

	Before Instruction	After Instruction
W8	23FF	FC18
SR	0000	0000

26a

Another important form of addressing is indirect addressing. An indirect address is indicated by placing square brackets around the register to indicate the source or destination address where the actual value is found.

The following instruction uses indirect addressing

mov [W6], [W7].

Here the value at the address in W6 is copied to the address in W7.

Example: Copy the value at data memory address 0x802 to data memory address 0x806 using indirect addressing

before			after		
	0x4A	0x802		0x4A	0x802
0x0802	0x67	0x803	0x0802	0x67	0x803
W6	0x01	0x804	W6	0x01	0x804
0x0806	0xB2	0x805	0x0806	0xB2	0x805
W7	0x20	0x806	W7	0x9A	0x806
	0x5C	0x807		0x67	0x807

A useful form of indirect addressing uses an offset from a base address in a working register to determine the effective address

Example: Copy the word in register W0 to the address two words after the address in W6

```
mov W0, [W6+0x2]
```

before

	0x4A
0x4321	0x67
W0	0x01
	0xB2
0x0802	0x20
W6	0x5C

after

	0x4A
0x4321	0x67
W0	0x01
	0xB2
0x0802	0x21
W6	0x43

The offset can be in the range -1024 to 1022

(For the byte version of this instruction the range is -512 to 511.)

MOV

Move [Ws with offset] to Wnd

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
	X	X	X	X	X	X

Syntax: {label:} MOV{.B} [Ws + Slit10], Wnd

Operands: Ws ∈ [W0 ... W15]
Slit10 ∈ [-512 ... 511] for byte operation
Slit10 ∈ [-1024 ... 1022] (even only) for word operation
Wnd ∈ [W0 ... W15]

Operation: [Ws + Slit10] →Wnd

Status Affected: None

Encoding:

1001	0kkk	kBkk	kddd	dkkk	ssss

Description: The contents of [Ws + Slit10] are loaded into Wnd. In Word mode, the range of Slit10 is increased to [-1024 ... 1022] and Slit10 must be even to maintain word address alignment. Register indirect addressing must be used for the source, and direct addressing must be used for Wnd.

The 'k' bits specify the value of the literal.
The 'B' bit selects byte or word operation ('0' for word, '1' for byte).
The 'd' bits select the destination register.
The 's' bits select the source register.

Note 1: The extension `.B` in the instruction denotes a byte move rather than a word move. You may use a `.W` extension to denote a word move, but it is not required.

2: In Byte mode, the range of Slit10 is not reduced as specified in **Section 4.6 "Using 10-bit Literal Operands"**, since the literal represents an address offset from Ws.

Words: 1

Cycles: 1(1)

Note 1: In dsPIC33E and PIC24E devices, the listed cycle count does not apply to read and read-modify-write operations on non-CPU Special Function Registers. For more details, see **Note 3** in **Section 3.2.1 "Multi-Cycle Instructions"**.

Example 1:

```
MOV.B   [W8+0x13], W10  ; load W10 with [W8+0x13]
                        ; (Byte mode)
```

	Before Instruction		After Instruction
W8	1008	W8	1008
W10	4009	W10	4033
Data 101A	3312	Data 101A	3312
SR	0000	SR	0000

Example 2:
```
MOV    [W4+0x3E8], W2  ; load W2 with [W4+0x3E8]
                       ; (Word mode)
```

	Before Instruction
W2	9088
W4	0800
Data 0BE8	5634
SR	0000

	After Instruction
W2	5634
W4	0800
Data 0BE8	5634
SR	0000

MOV

Move Wns to [Wd with offset]

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
	X	X	X	X	X	X

Syntax: {label:} MOV{.B} Wns, [Wd + Slit10]

Operands:
Wns ∈ [W0 ... W15]
Slit10 ∈ [-512 ... 511] in Byte mode
Slit10 ∈ [-1024 ... 1022] (even only) in Word mode
Wd ∈ [W0 ... W15]

Operation: (Wns) →[Wd + Slit10]

Status Affected: None

Encoding:

1001	1kkk	kBkk	kddd	dkkk	ssss

Description: The contents of Wns are stored to [Wd + Slit10]. In Word mode, the range of Slit10 is increased to [-1024 ... 1022] and Slit10 must be even to maintain word address alignment. Register direct addressing must be used for Wns, and indirect addressing must be used for the destination.

The 'k' bits specify the value of the literal.
The 'B' bit selects byte or word operation ('0' for word, '1' for byte).
The 'd' bits select the destination register.
The 's' bits select the source register.

Note 1: The extension .B in the instruction denotes a byte move rather than a word move. You may use a .W extension to denote a word move, but it is not required.

2: In Byte mode, the range of Slit10 is not reduced as specified in **Section 4.6 "Using 10-bit Literal Operands"**, since the literal represents an address offset from Wd.

Words: 1

Cycles: 1

Example 1:
```
MOV.B  W0, [W1+0x7]    ; store W0 to [W1+0x7]
                       ; (Byte mode)
```

	Before Instruction
W0	9015
W1	1800
Data 1806	2345
SR	0000

	After Instruction
W0	9015
W1	1800
Data 1806	1545
SR	0000

Arithmetic

The most common arithmetic operation in any microcontroller is addition. Addition is not only used in processing data but is also used for operations such as computing offsets in addressing.

The PIC24 implements both three and two operand versions of addition (and subtraction). The three operand version allows separate registers for each of the two numbers to be added and the result. As with the mov instruction various addressing modes are supported with variants of these instructions that will operate on bytes

Example: Add the contents of w0 to w1 and place the result in w3.

```
add w0, w1, w3
```

This instruction uses register direct addressing with the two source operands preceding the destination operand.

The actual operation is (w0) + (w1) → w3

w0	0x1234	w0	0x1234
w1	0x5678	w1	0x5678
w3	0x9ABC	w3	0x68AC

Example 2:

```
ADD     W3, #0x6, [--W4]   ; Add W3 and 6 (Word mode)
                           ; Store the result in [--W4]
```

	Before Instruction	After Instruction
W3	6006	6006
W4	1000	0FFE
Data 0FFE	DDEE	600C
Data 1000	DDEE	DDEE
SR	0000	0000

ADD

Add Wb to Ws

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
	X	X	X	X	X	X

Syntax:

{label:}	ADD{.B}	Wb,	Ws,	Wd
			[Ws],	[Wd]
			[Ws++],	[Wd++]
			[Ws--],	[Wd--]
			[++Ws],	[++Wd]
			[--Ws],	[--Wd]

Operands:
Wb ∈ [W0 ... W15]
Ws ∈ [W0 ... W15]
Wd ∈ [W0 ... W15]

Operation: (Wb) + (Ws) →Wd

Status Affected: DC, N, OV, Z, C

Encoding:

0100	0www	wBqq	qddd	dppp	ssss

Description: Add the contents of the source register Ws and the contents of the base register Wb, and place the result in the destination register Wd. Register direct addressing must be used for Wb. Either register direct or indirect addressing may be used for Ws and Wd.

The 'w' bits select the address of the base register.
The 'B' bit selects byte or word operation ('0' for word, '1' for byte).
The 'q' bits select the destination Address mode.
The 'd' bits select the destination register.
The 'p' bits select the source Address mode.
The 's' bits select the source register.

Note: The extension `.B` in the instruction denotes a byte operation rathjer than a word operation. You may use a `.W` extension to denote a word operation, but it is not required.

Words: 1

Cycles: 1(1)

Note 1: In dsPIC33E and PIC24E devices, the listed cycle count does not apply to read and read-modify-write operations on non-CPU Special Function Registers. For more details, see **Note 3** in **Section 3.2.1 "Multi-Cycle Instructions"**.

30d

Example 1:
```
ADD.B    W5, W6, W7    ; Add W5 to W6, store result in W7
                       ; (Byte mode)
```

	Before Instruction	After Instruction
W5	AB00	AB00
W6	0030	0030
W7	FFFF	FF30
SR	0000	0000

Example 2:
```
ADD      W5, W6, W7    ; Add W5 to W6, store result in W7
                       ; (Word mode)
```

	Before Instruction	After Instruction
W5	AB00	AB00
W6	0030	0030
W7	FFFF	AB30
SR	0000	0008 (N = 1)

ADD

Add Accumulators

Implemented in:	PIC24F	PIC24H	PIC24E	dsPIC30F	dsPIC33F	dsPIC33E
				X	X	X

Syntax: {label:} ADD Acc

Operands: Acc ∈ [A,B]

Operation:
If (Acc = A):
(ACCA) + (ACCB) →ACCA
Else:
(ACCA) + (ACCB) →ACCB

Status Affected: OA, OB, OAB, SA, SB, SAB

Encoding:

1100	1011	A000	0000	0000	0000

Description: Add the contents of Accumulator A to the contents of Accumulator B and place the result in the selected accumulator. This instruction performs a 40-bit addition.

The 'A' bit specifies the destination accumulator.

Words: 1

Cycles: 1

Example 1:
```
ADD      A             ; Add ACCB to ACCA
```

	Before Instruction	After Instruction
ACCA	00 0022 3300	00 1855 7858
ACCB	00 1833 4558	00 1833 4558
SR	0000	0000

30b

>xc16-objdump -d newmainXC16.o

```
Disassembly of section .text:

00000000 <_main>:
   0:     06 00 fa        lnk         #0x6
   2:     50 f8 2f        mov.w       #0xff85, w0
   4:     00 0f 78        mov.w       w0, [w14]
   6:     e0 02 20        mov.w       #0x2e, w0
   8:     10 07 98        mov.w       w0, [w14+2]
   a:     1e 00 90        mov.w       [w14+2], w0
   c:     1e 00 40        add.w       w0, [w14], w0
   e:     20 07 98        mov.w       w0, [w14+4]
  10:     2e 00 90        mov.w       [w14+4], w0
  12:     00 00 88        mov.w       w0, 0x0
  14:     00 00 eb        clr.w       w0
  16:     00 80 fa        ulnk
  18:     00 00 06        return
```