Key Objectives

- Familiarity of 16-bit Architecture
- Knowledge of 16-bit Instruction Set
- Learning how to program using the 16-bit Assembly Language

Session Agenda

- 16-bit Architecture Basics
- Some 16-bit Assembly Instructions
- I/O Port Handling
  - **Hands-on Lab #1 Follow along**
    - **Basic Assembly language setup**
    - **Light LED on I/O Port**
- Complete 16-bit Assembly Instructions
- Addressing Modes
  - **Hands-on Lab #2**
    - **Use Loop Instructions to Blink LED**
- Interrupts and Timers
  - **Hands-on Lab #3**
    - **Use Interrupt to Blink LED**
- Program Memory Access
  - **Optional Hands-on Lab #4**
    - **Use PSV to transmit Serial String "Hello World"**

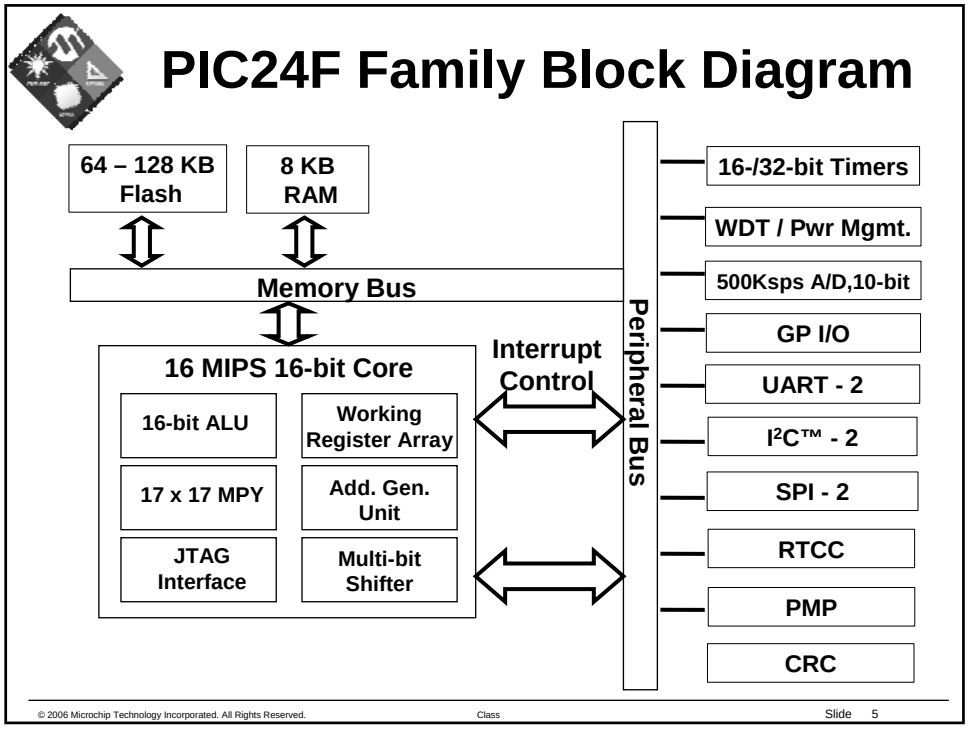
 Class Slide 3

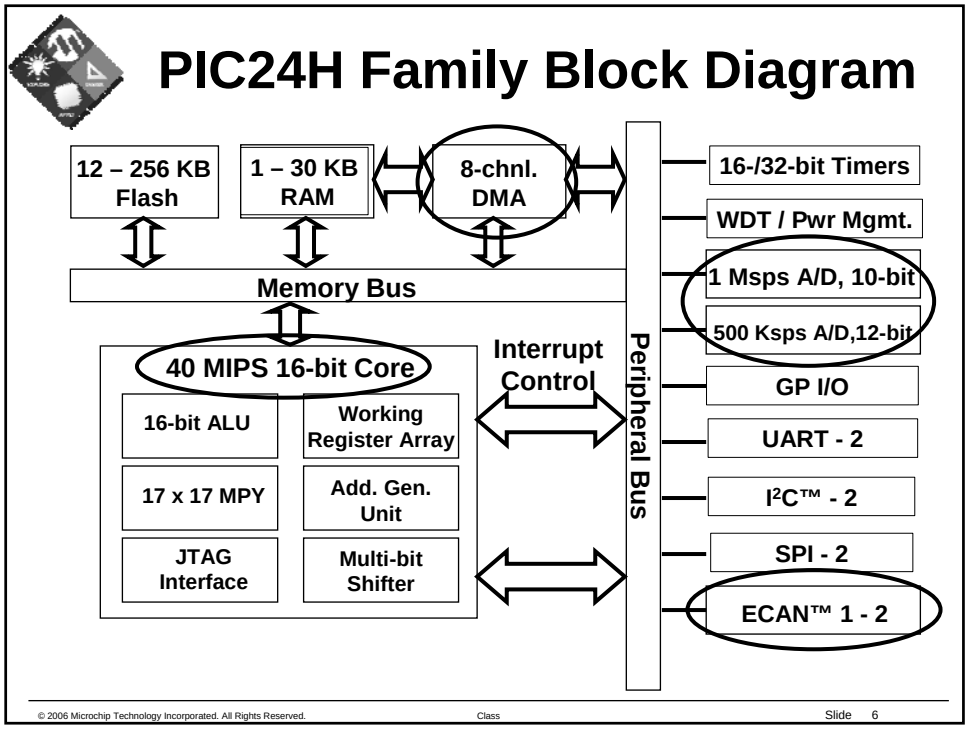
16-bit Product Families Overview

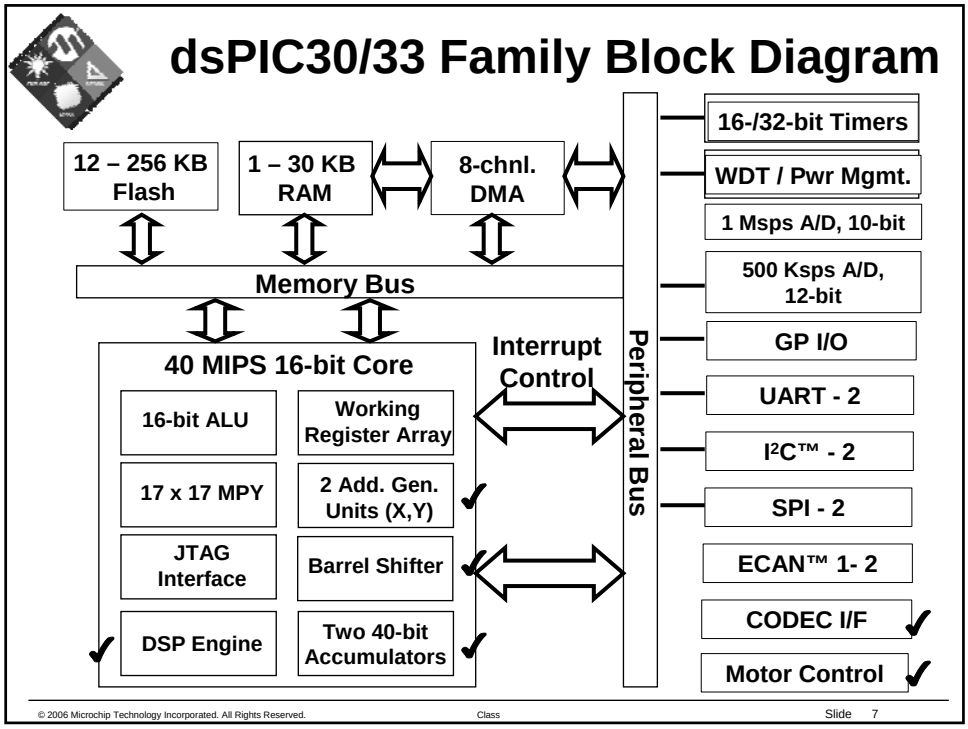
PIC24F **PIC24H** **dsPIC30F** **dsPIC33F**

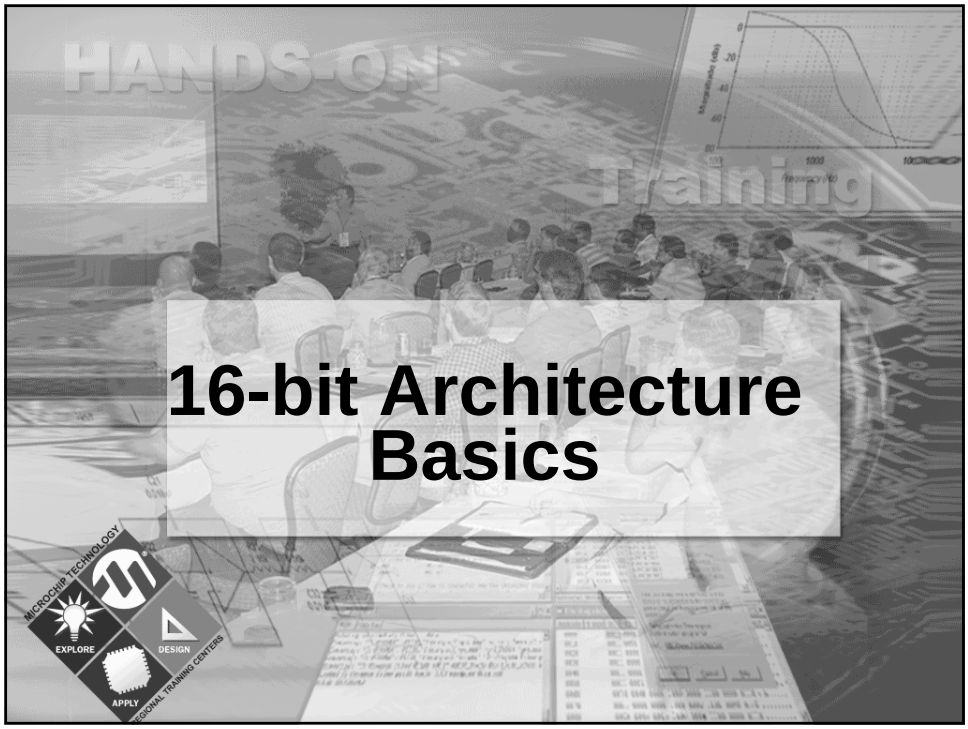
- ✓ Same Basic Architecture
- ✓ Same 24-bit Flash Program Memory
- ✓ Same 16-bit Data Memory
- ✓ Instruction Set optimized for C efficiency
- ✓ Same Deterministic Interrupt System
- ✓ Flexible system clock features
- ✓ DSP Performance when you need it
- ✓ Advanced Peripherals
- ✓ Easy Code Migration

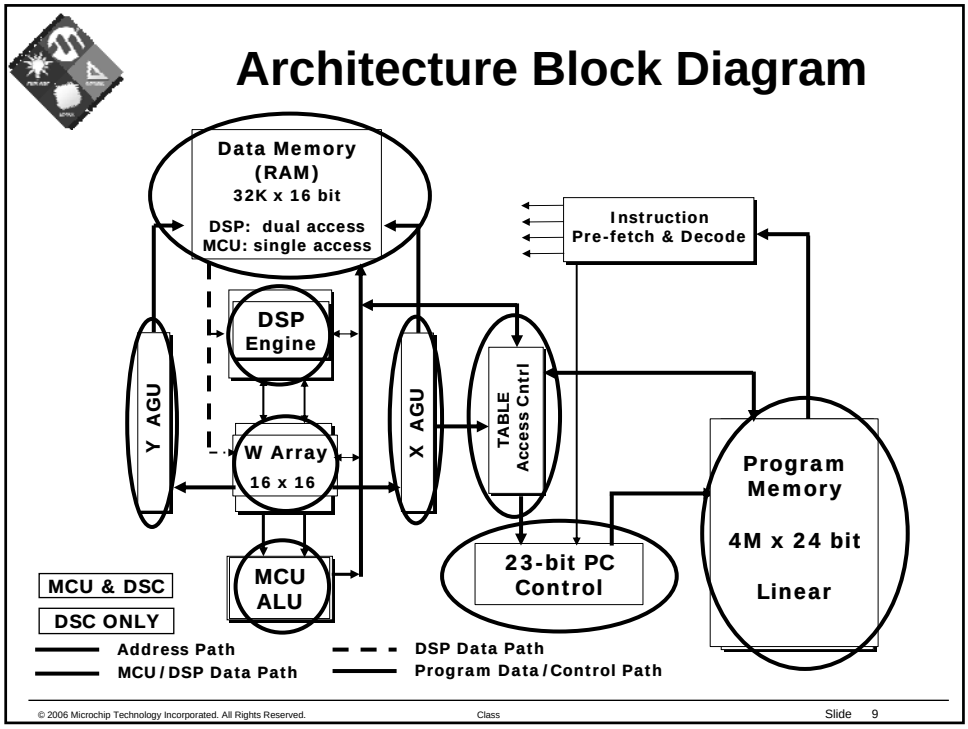
 Class Slide 4

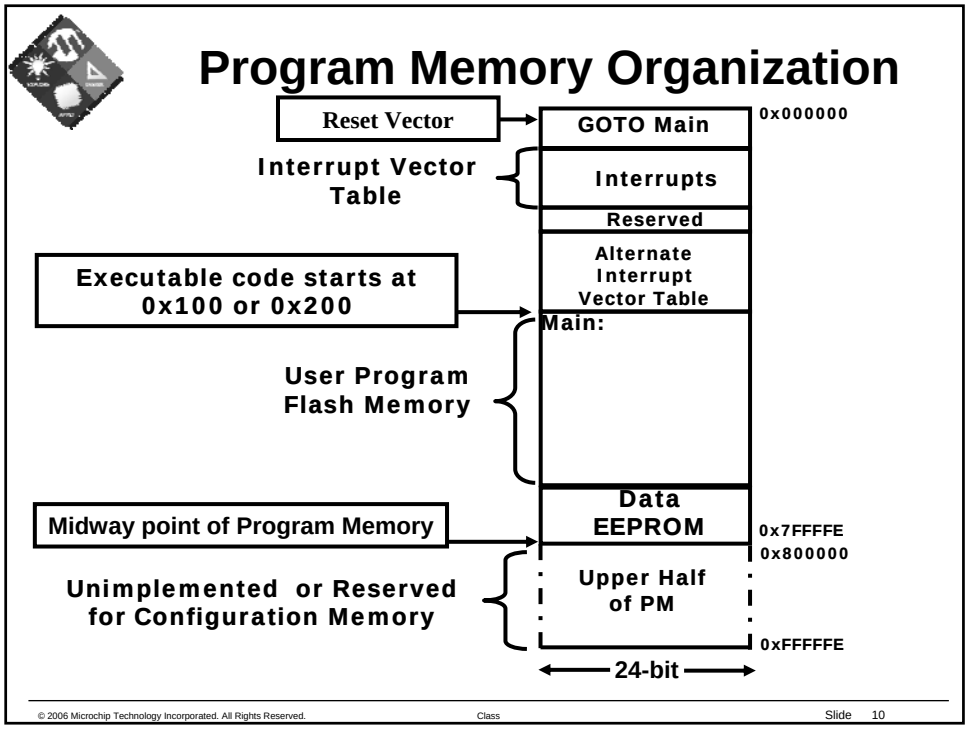

PIC24F Family Block Diagram
64 – 128 KB Flash
8 KB RAM
Memory Bus
16 MIPS 16-bit Core
16-bit ALU
Working Register Array
17 x 17 MPY
Add. Gen. Unit
JTAG Interface
Multi-bit Shifter
Interrupt Control
Peripheral Bus
16-/32-bit Timers
WDT / Pwr Mgmt.
500Ksps A/D,10-bit
GP I/O
UART - 2
I2C™ - 2
SPI - 2
RTCC
PMP
CRC
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 5


PIC24H Family Block Diagram
12 – 256 KB Flash
1 – 30 KB RAM
8-chnl. DMA
Memory Bus
40 MIPS 16-bit Core
16-bit ALU
Working Register Array
17 x 17 MPY
Add. Gen. Unit
JTAG Interface
Multi-bit Shifter
Interrupt Control
Peripheral Bus
16-/32-bit Timers
WDT / Pwr Mgmt.
1 Msps A/D, 10-bit
500 Ksps A/D,12-bit
GP I/O
UART - 2
I2C™ - 2
SPI - 2
ECAN™ 1 - 2
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 6

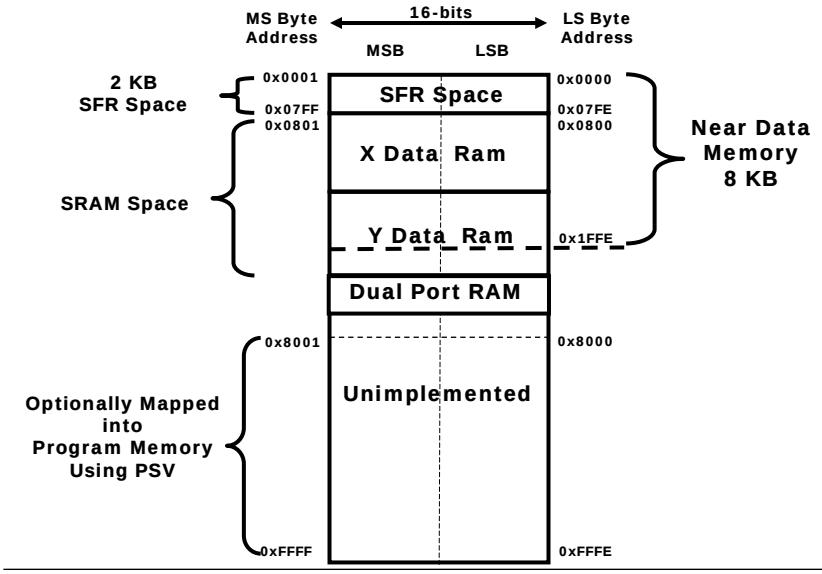

dsPIC30/33 Family Block Diagram
12 – 256 KB Flash
1 – 30 KB RAM
8-chnl. DMA
Memory Bus
40 MIPS 16-bit Core
16-bit ALU
Working Register Array
17 x 17 MPY
2 Add. Gen. Units (X,Y)
JTAG Interface
Barrel Shifter
DSP Engine
Two 40-bit Accumulators
Interrupt Control
Peripheral Bus
16-/32-bit Timers
WDT / Pwr Mgmt.
1 Msps A/D, 10-bit
500 Ksps A/D, 12-bit
GP I/O
UART - 2
I²C™ - 2
SPI - 2
ECAN™ 1- 2
CODEC I/F
Motor Control
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 7


HANDS-ON
Training
16-bit Architecture Basics
MICROCHIP TECHNOLOGY
EXPLORE
DESIGN
APPLY
REGIONAL TRAINING CENTERS


Architecture Block Diagram
Data Memory (RAM)
32K x 16 bit
DSP: dual access
MCU: single access
Instruction Pre-fetch & Decode
DSP Engine
Y AGU
W Array
16 x 16
X AGU
TABLE Access Cntrl
Program Memory
4M x 24 bit
Linear
MCU ALU
23-bit PC Control
MCU & DSC
DSC ONLY
Address Path
MCU/DSP Data Path
DSP Data Path
Program Data/Control Path
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 9

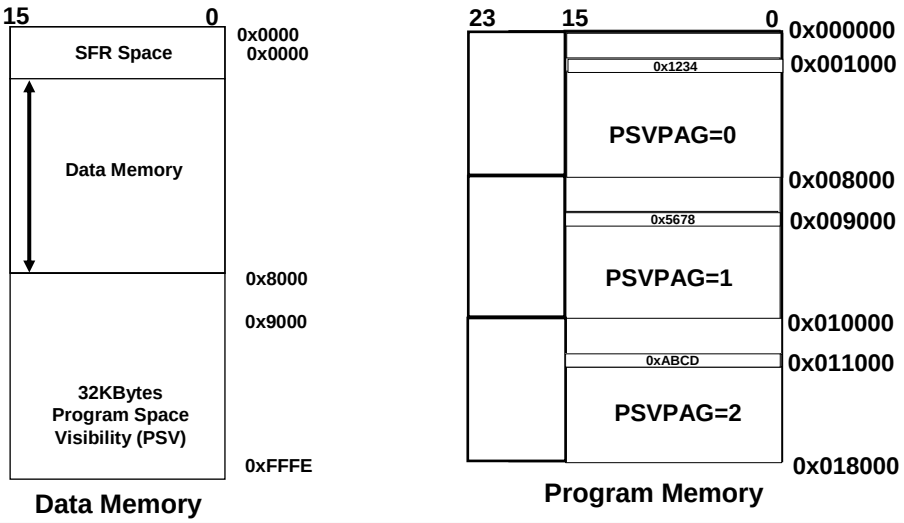

Program Memory Organization
Reset Vector
GOTO Main
0x000000
Interrupt Vector Table
Interrupts
Reserved
Alternate Interrupt Vector Table
Executable code starts at 0x100 or 0x200
Main:
User Program Flash Memory
Midway point of Program Memory
Data EEPROM
0x7FFFFE
0x800000
Unimplemented or Reserved for Configuration Memory
Upper Half of PM
0xFFFFFE
24-bit
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 10

Data Memory Organization*

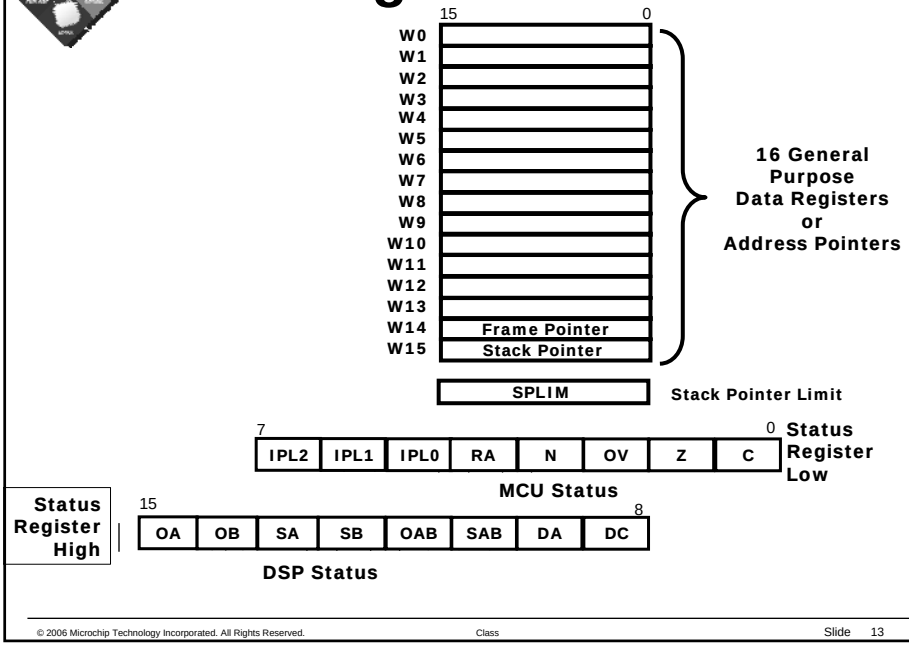


Program Space Visibility

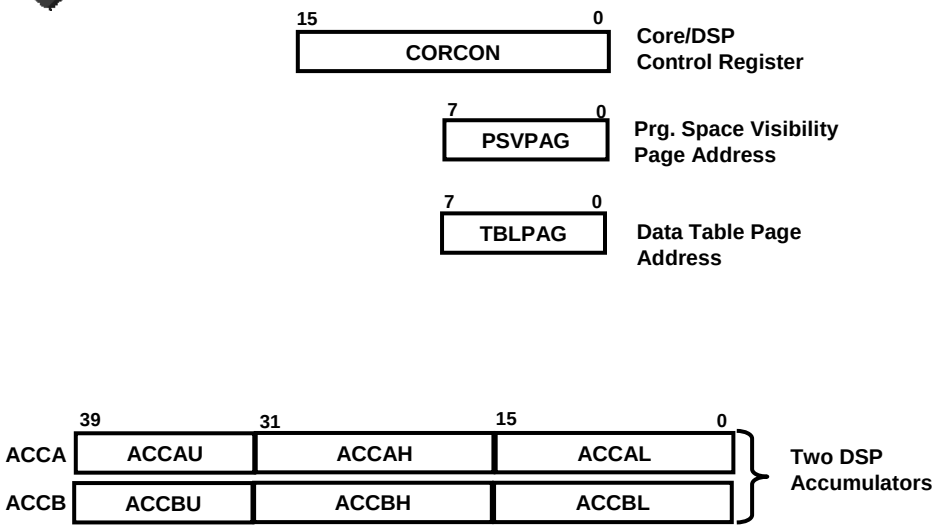
- Program Space Visibility Enabled when CORCON<PSV> = 1
- PSVPAG register maps 32Kb program memory segment into data memory



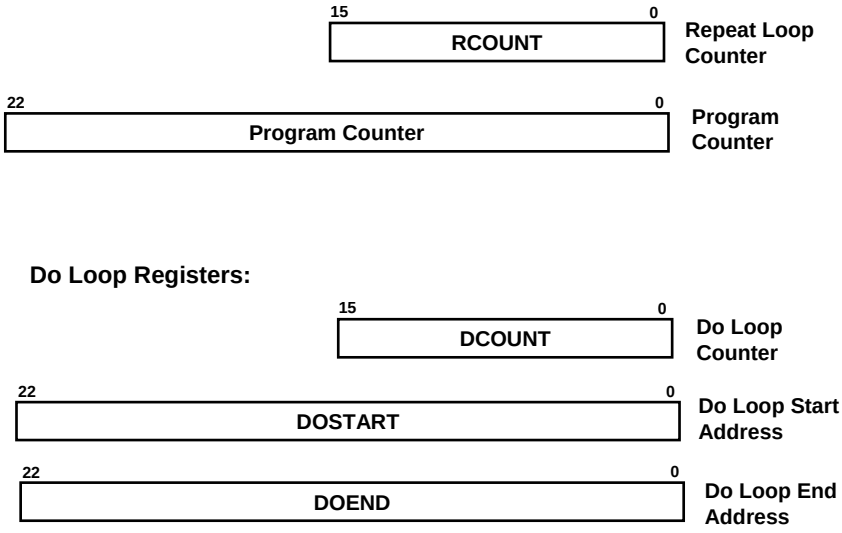
Programmers Model



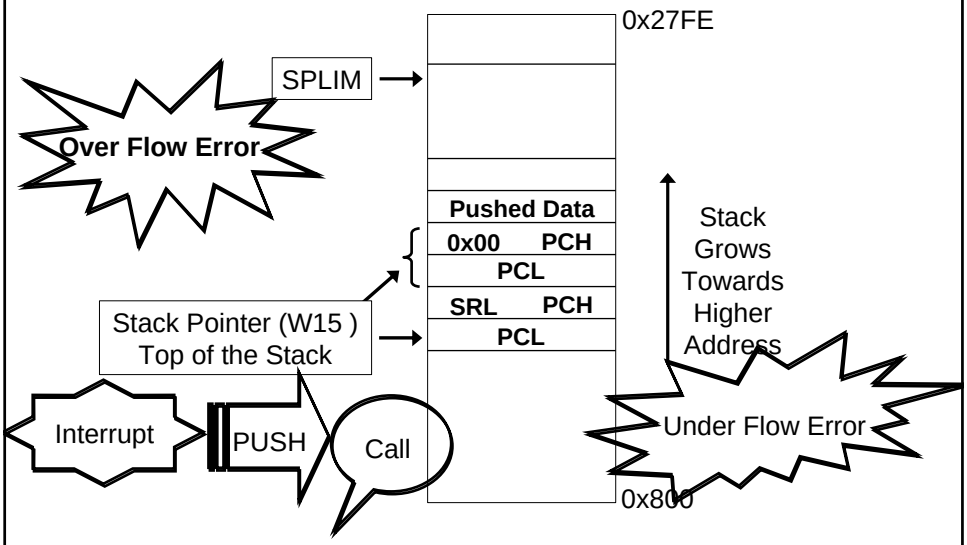
Programmers Model (contd.)



Programmers Model(contd.)

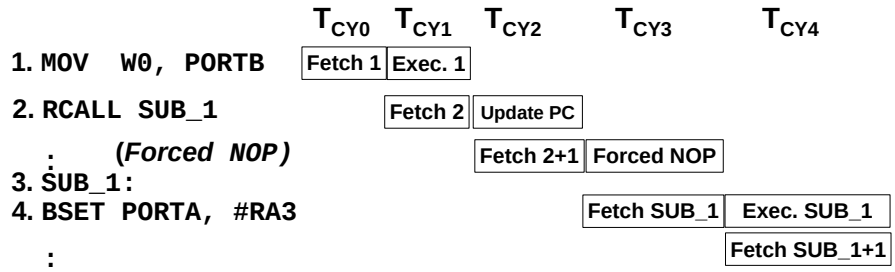


Software Stack in Data RAM



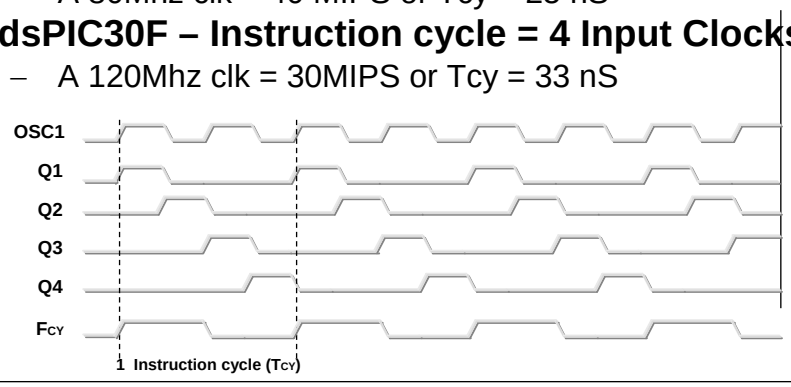
Instruction Pipeline

- Allows overlap of execution and fetch
- Makes single cycle execution
- Program branches (e.g. GOTO, CALL) take two cycles



Clocking Scheme

- Instruction cycle = 2 Input Clocks
- PIC24F
  - A 32Mhz clk = 16MIPS or Tcy = 62.5 nS
- PIC24H/dsPIC33
  - A 80Mhz clk = 40 MIPS or Tcy = 25 nS
- dsPIC30F – Instruction cycle = 4 Input Clocks
  - A 120Mhz clk = 30MIPS or Tcy = 33 nS



Session Agenda

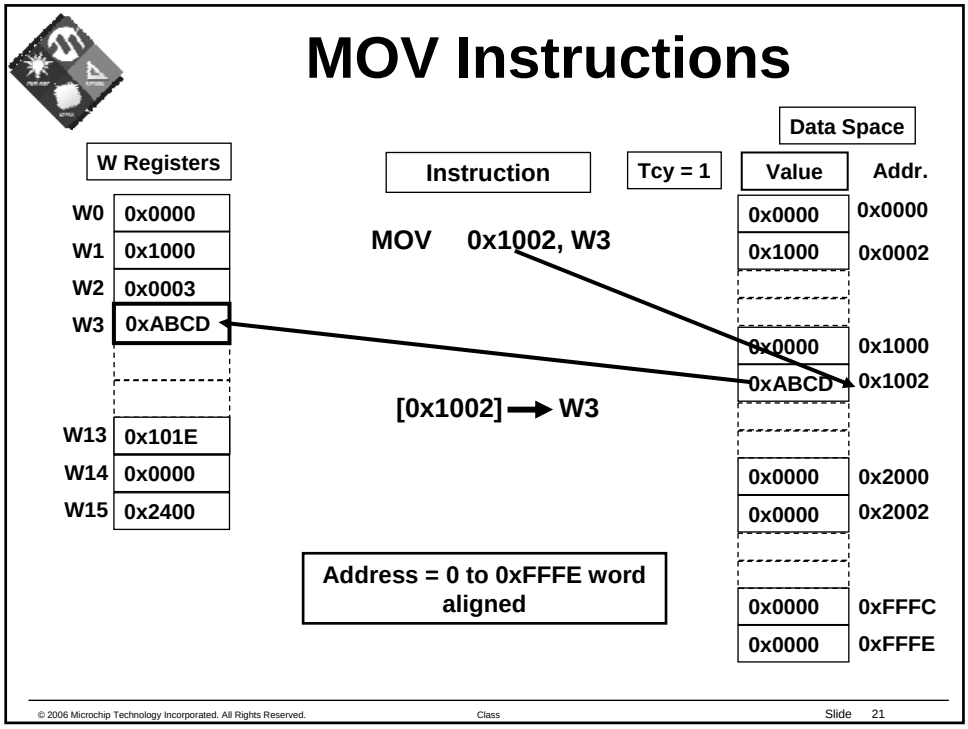
- 16-bit Architecture Basics
- Some 16-bit Assembly Instructions
- I/O Port Handling
  - **Hands-on Lab #1 Follow along**
    - **Basic Assembly language setup**
    - **Light LED on I/O Port**
- Complete 16-bit Assembly Instructions
  - **Hands-on Lab #2**
    - **Use Loop Instructions to Blink LED**
- Addressing Modes
- Interrupts and Timer1
  - **Hands-on Lab #3**
    - **Use Interrupt to Blink LED**

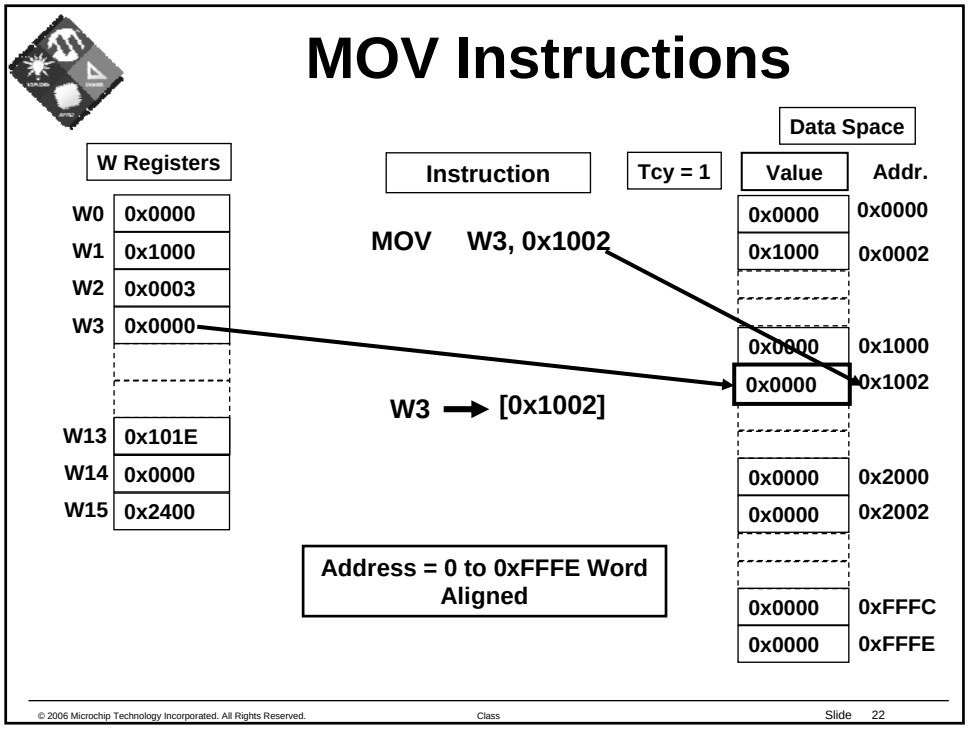
 Class Slide 19

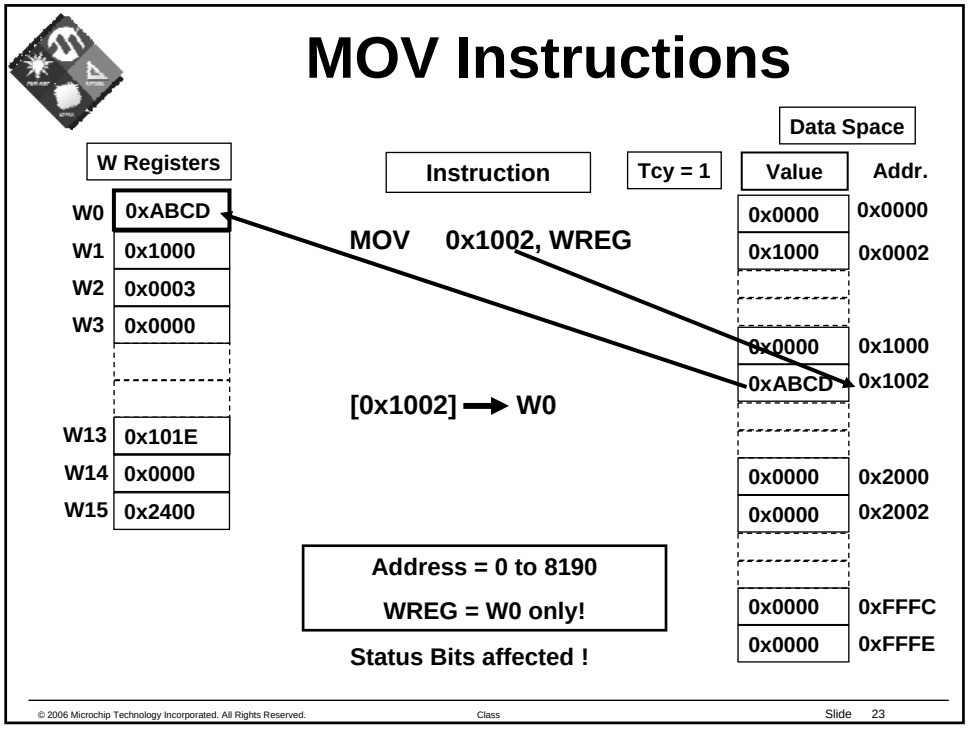
Instruction Set Functional Groups

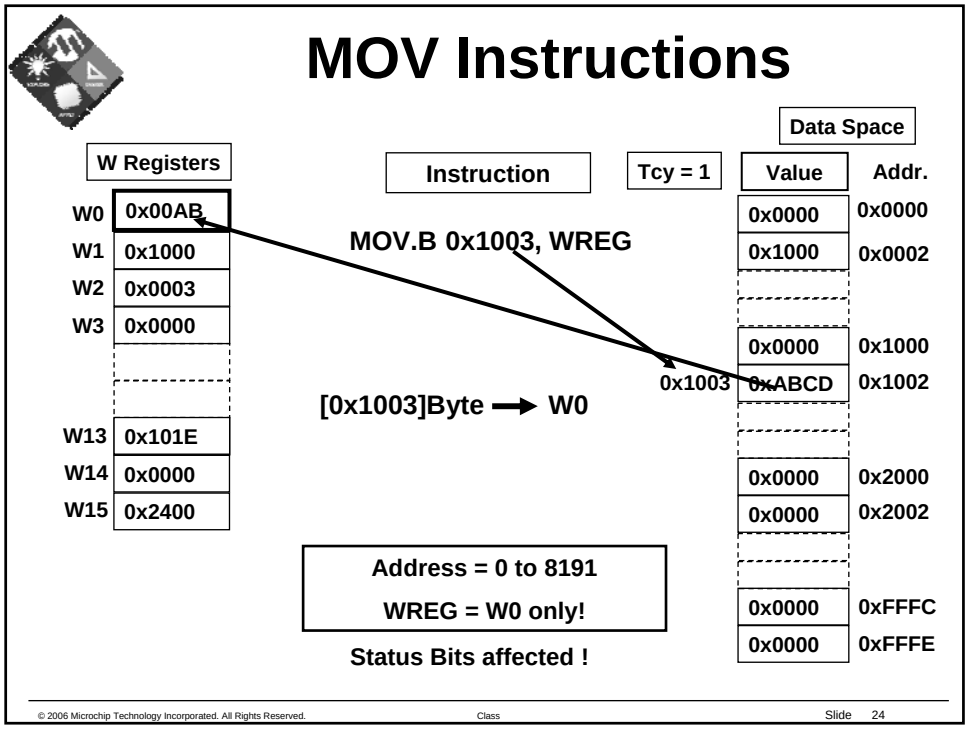
- **16-bit Instructions**
  - **Move Instructions**
  - **Bit Instructions**
    - Follow along LAB 1
  - **Math and Logic Instructions**
  - **Stack Control Instructions**
  - **Program Flow Control Instructions**
  - **CPU Control Instructions**
    - LAB 2

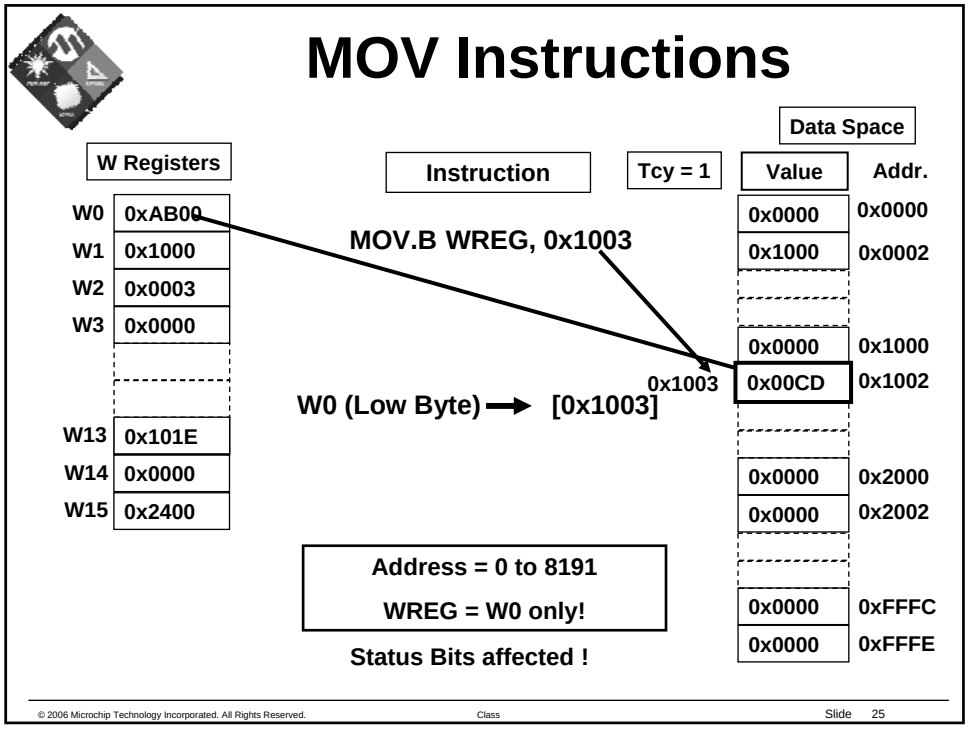
 Class Slide 20

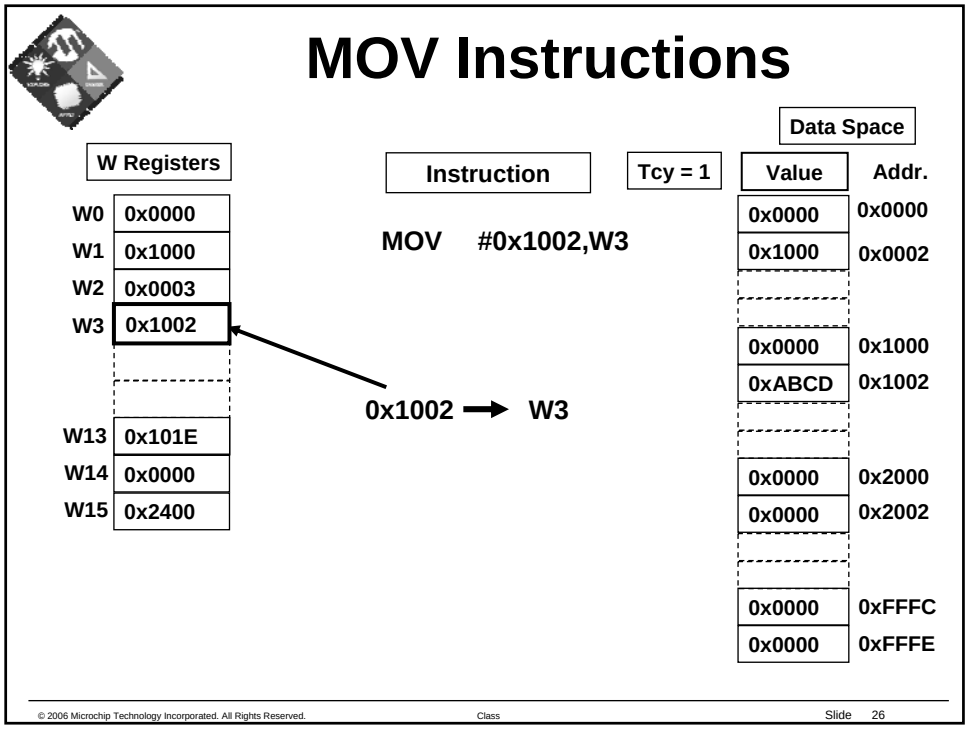

MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0xABCD
W13 0x101E
W14 0x0000
W15 0x2400
MOV 0x1002, W3
[0x1002] → W3
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Address = 0 to 0xFFFE word aligned
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 21

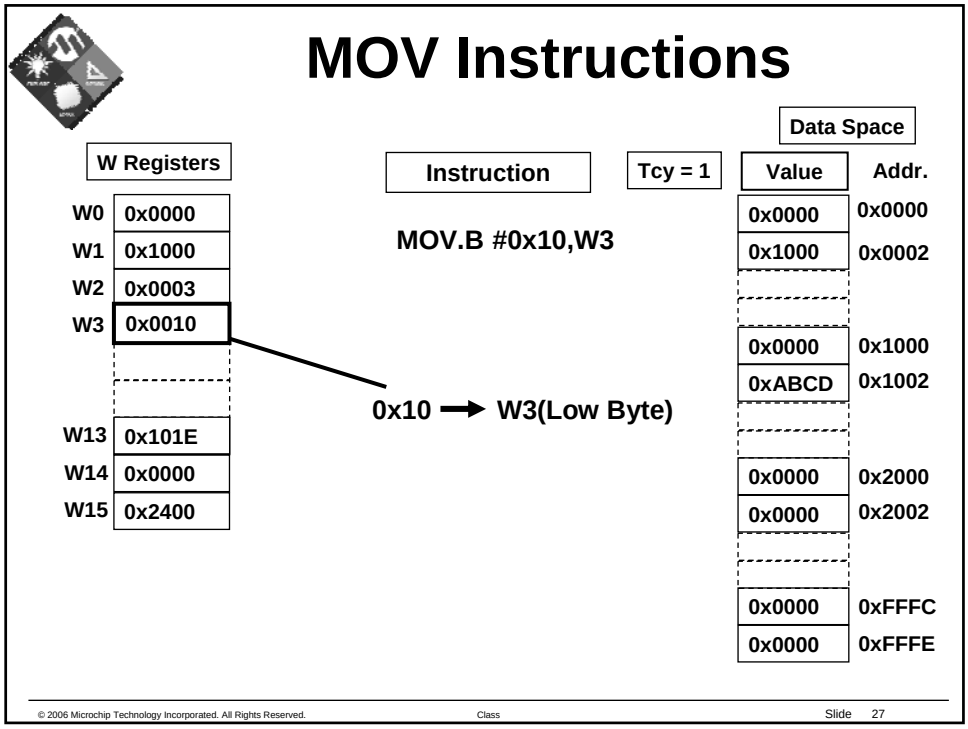

MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
MOV W3, 0x1002
W3 → [0x1002]
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0x0000 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Address = 0 to 0xFFFE Word Aligned
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 22

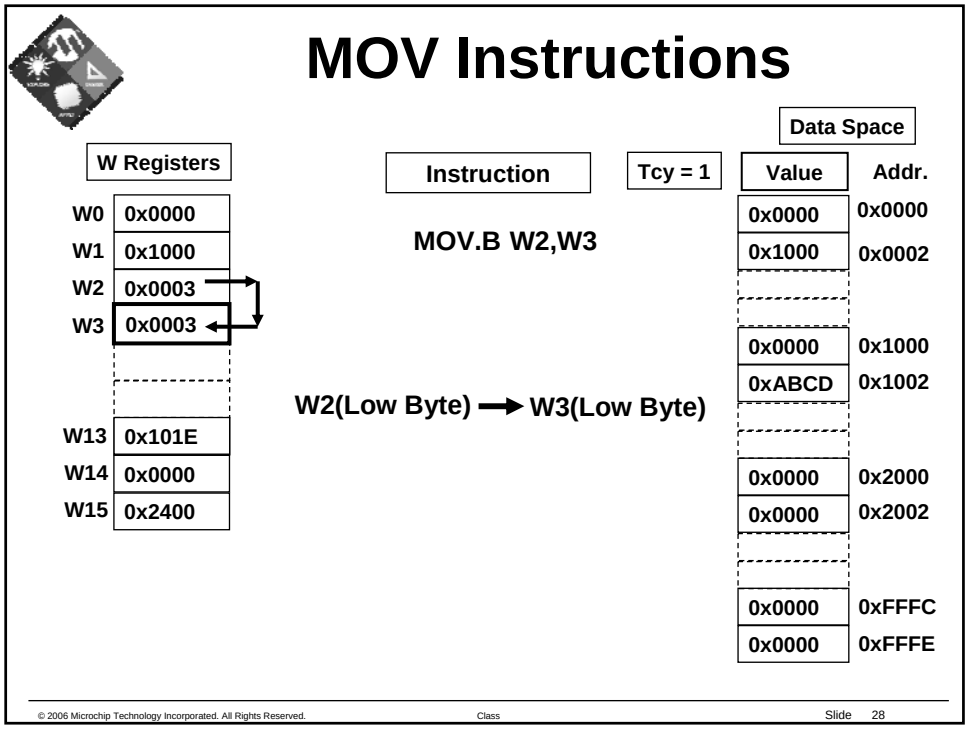

MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0xABCD
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
MOV 0x1002, WREG
[0x1002] → W0
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Address = 0 to 8190
WREG = W0 only!
Status Bits affected !
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 23


MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x00AB
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
MOV.B 0x1003, WREG
[0x1003]Byte → W0
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0x1003 0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Address = 0 to 8191
WREG = W0 only!
Status Bits affected !
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 24

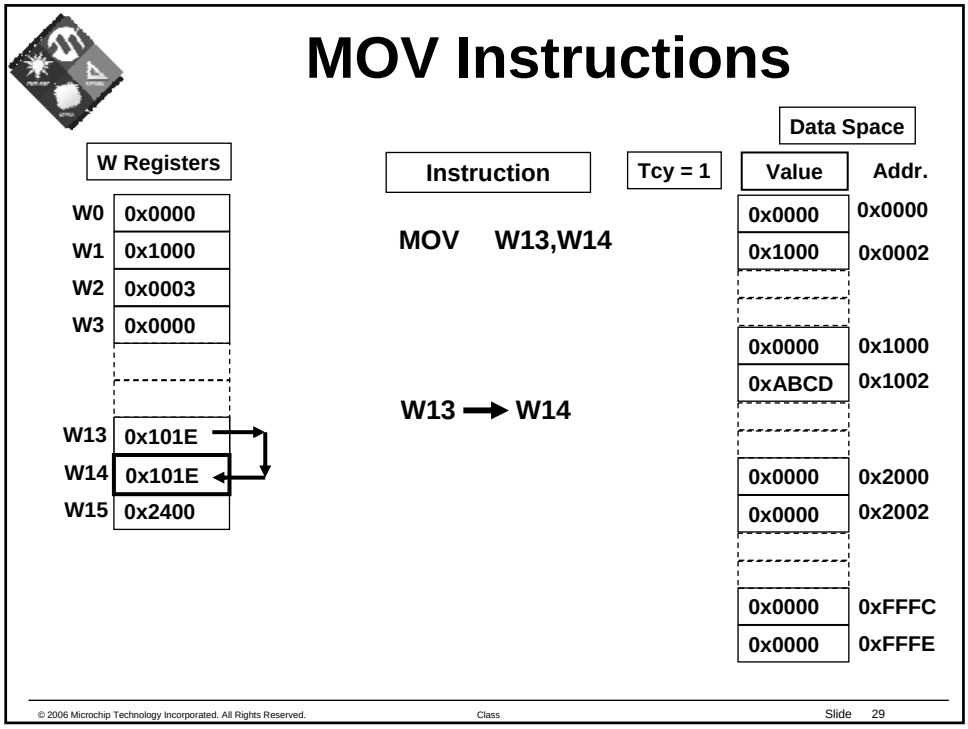

MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0xAB00
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
MOV.B WREG, 0x1003
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0x1003
0x00CD 0x1002
W0 (Low Byte) → [0x1003]
0x0000 0x2000
0x0000 0x2002
Address = 0 to 8191
WREG = W0 only!
0x0000 0xFFFC
0x0000 0xFFFE
Status Bits affected !
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 25


MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x1002
W13 0x101E
W14 0x0000
W15 0x2400
MOV #0x1002,W3
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x1002 → W3
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 26

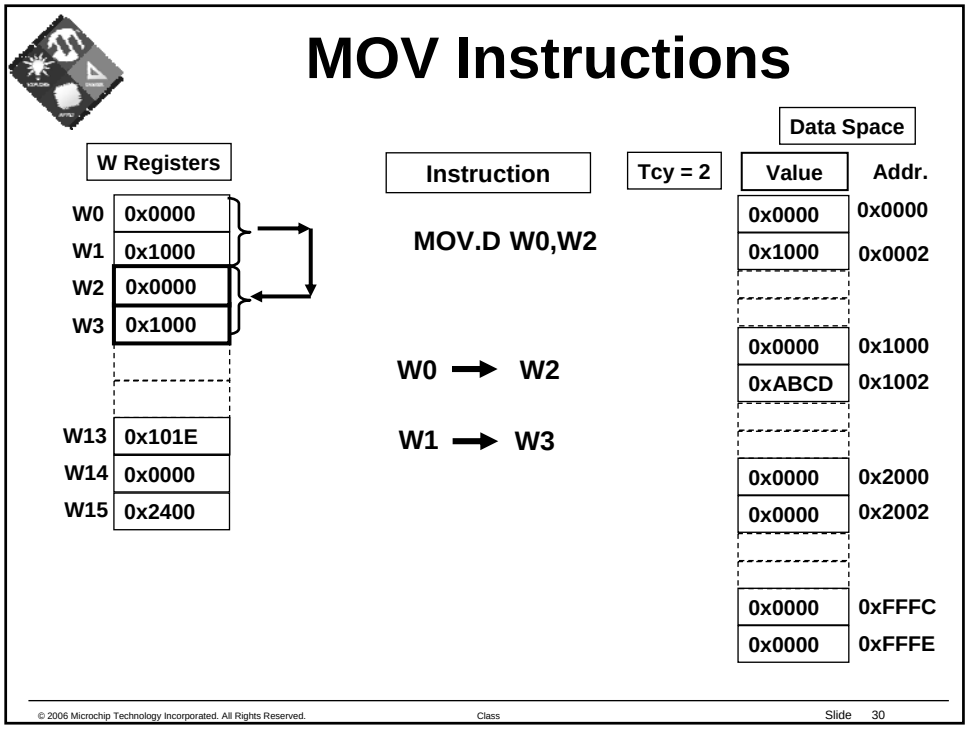

MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0010
W13 0x101E
W14 0x0000
W15 0x2400
MOV.B #0x10,W3
0x10 → W3(Low Byte)
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 27


MOV Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0003
W13 0x101E
W14 0x0000
W15 0x2400
MOV.B W2,W3
W2(Low Byte) → W3(Low Byte)
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 28

MOV Instructions



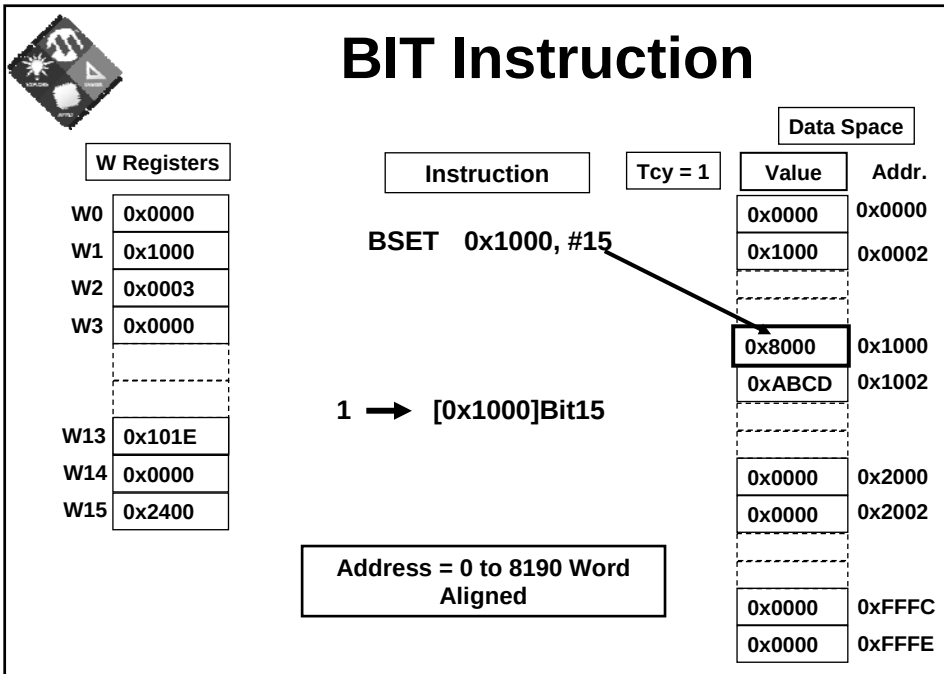
MOV Instructions

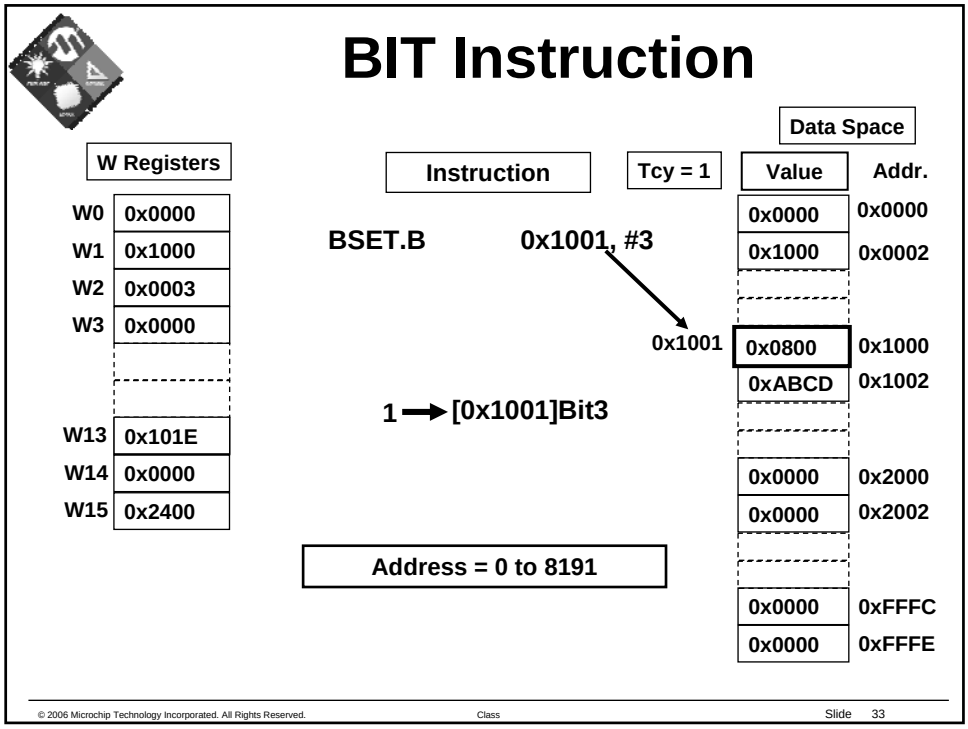


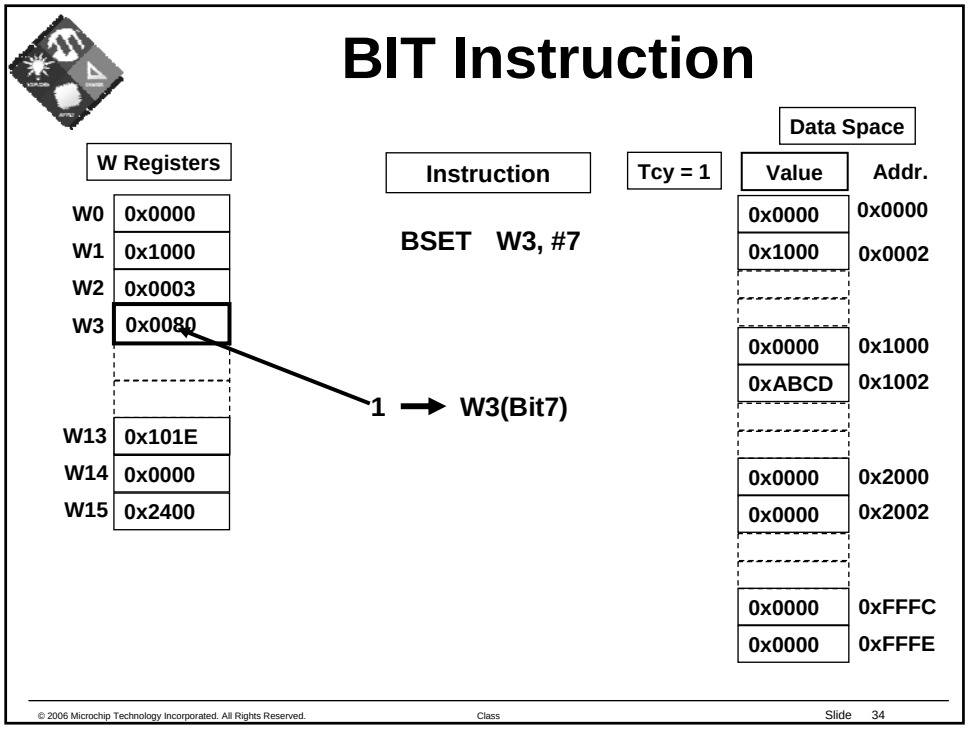
Instruction Set Functional Groups

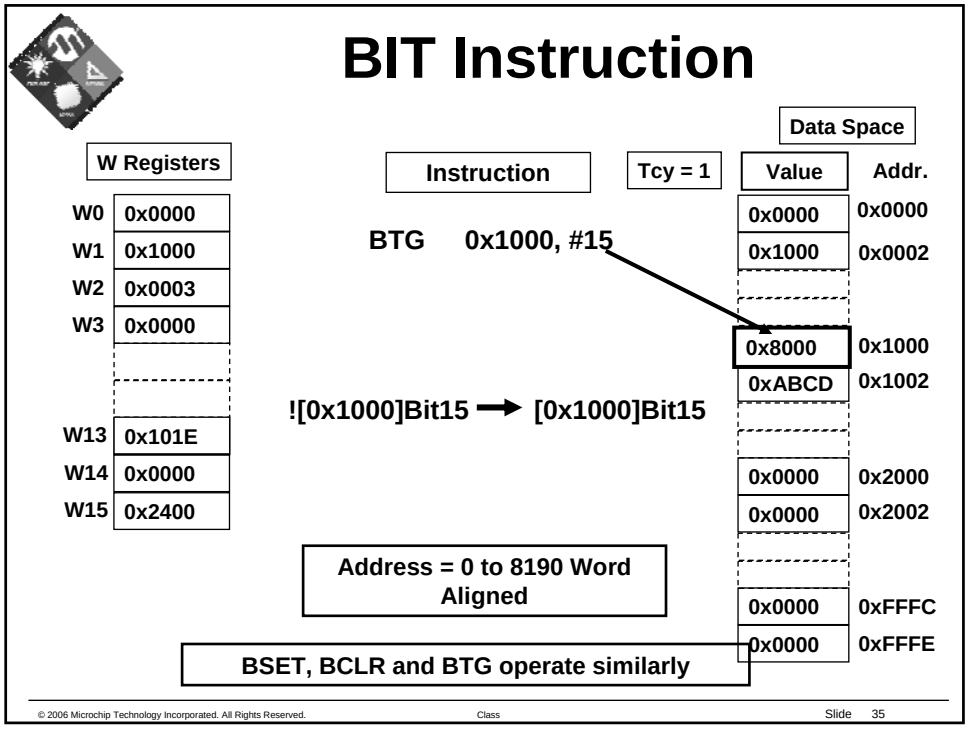
- 16-bit Instructions
  - Move Instructions
  - Bit Instructions
    - Follow along LAB 1
  - Math and Logic Instructions
  - Stack Control Instructions
  - Program Flow Control Instructions
  - CPU Control Instructions
    - LAB 2

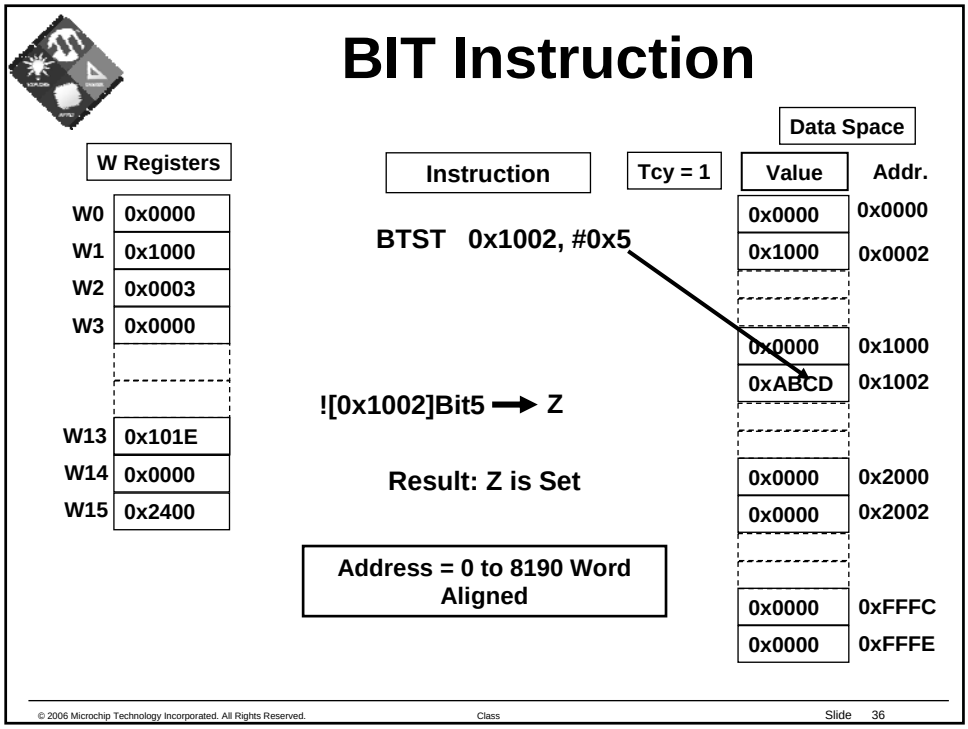
BIT Instruction



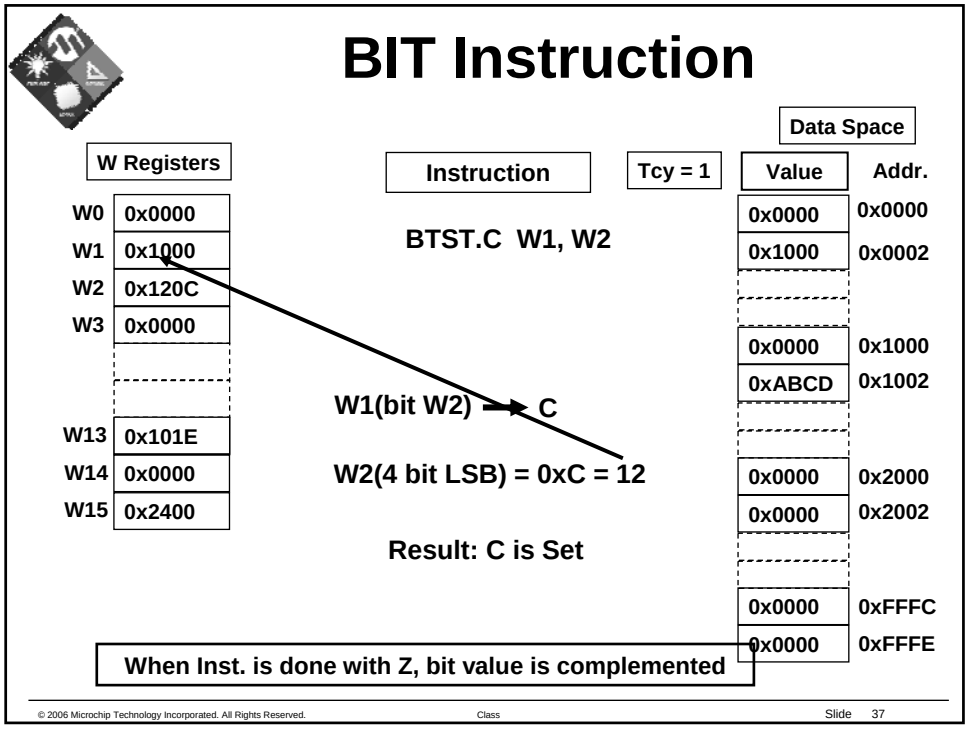

BIT Instruction
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
BSET.B 0x1001, #3
0x1001
1 → [0x1001]Bit3
Address = 0 to 8191
0x0000 0x0000
0x1000 0x0002
0x0800 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 33


BIT Instruction
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0080
W13 0x101E
W14 0x0000
W15 0x2400
BSET W3, #7
1 → W3(Bit7)
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 34

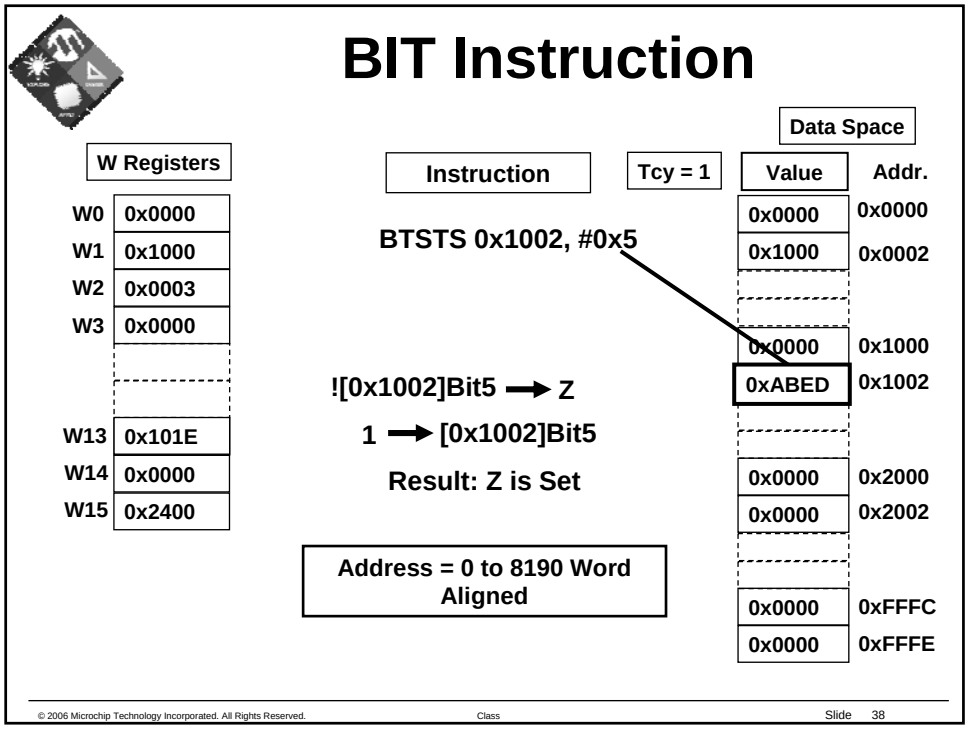

BIT Instruction
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
BTG 0x1000, #15
0x0000 0x0000
0x1000 0x0002
0x8000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
![0x1000]Bit15 → [0x1000]Bit15
Address = 0 to 8190 Word Aligned
BSET, BCLR and BTG operate similarly
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 35


BIT Instruction
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x0000
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
BTST 0x1002, #0x5
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
![0x1002]Bit5 → Z
Result: Z is Set
Address = 0 to 8190 Word Aligned
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 36

BIT Instruction



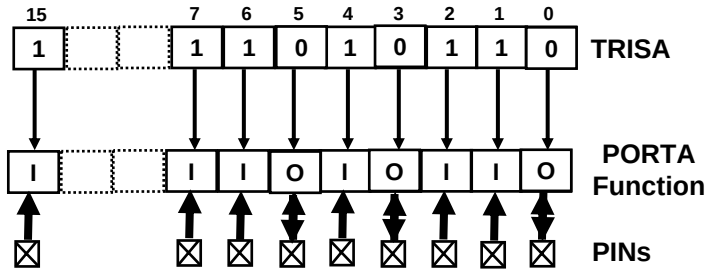
BIT Instruction



Instruction Set Functional Groups

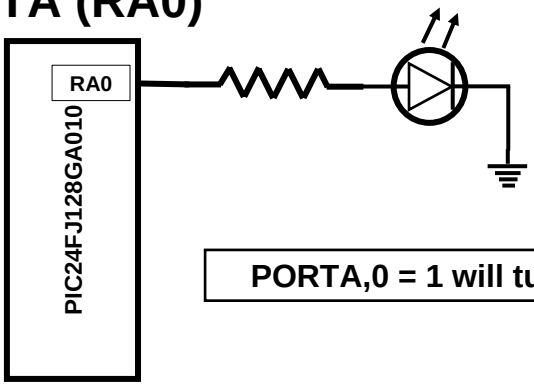
- **16-bit Instructions**
  - **Move Instructions**
  - **Bit Instructions**
    - Follow along LAB 1
  - **Math and Logic Instructions**
  - **Stack Control Instructions**
  - **Program Flow Control Instructions**
  - **CPU Control Instructions**
    - LAB 2

I/O PORT

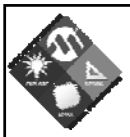


Lab 1: The Task

- Make PORTA, bit 0 as output
- Turn on LED connected to bit 0 of PORTA (RA0)



PORTA,0 = 1 will turn on LED



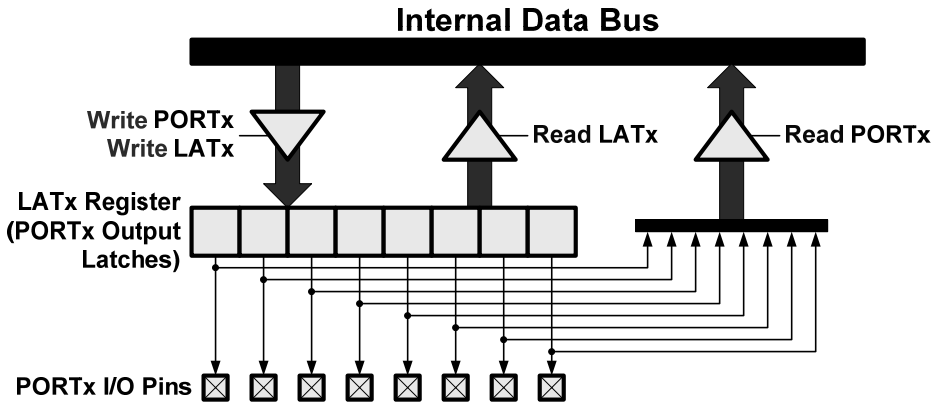
LAB1

- **Open Lab1.mcw workspace in MPLAB**
  - C:\rtc\103ASP\Lab1.mcw
- **Follow along and edit code:**
  - Setup Stack Pointer and Stack Limit
  - Setup TRISA
  - Bit Set PORTA, bit 0
- **Compile Code**
- **Program Part on Explorer-16 using ICD2**
- **Run Code using ICD2**
- **See the LED light up!!!**

More Than one LED lights up

- **You may notice:**
  - More than One LED Lights up when RUN is executes
  - WHY?

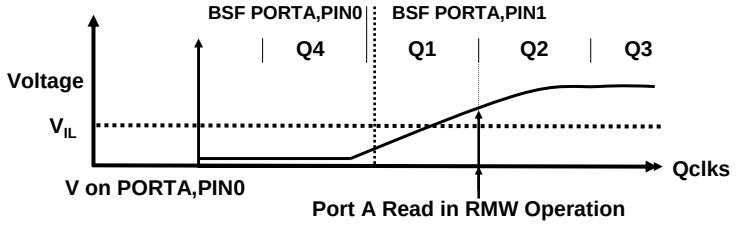
Digital I/O Ports – 16-bit



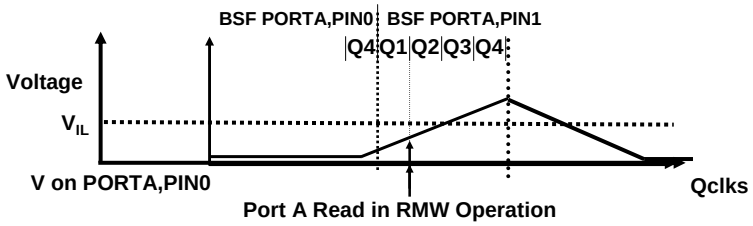

```
BSET   PORTA,#0      ;Turn LED @ PORTA bit 0 ON
BSET   PORTA,#1      ;Turn LED @ PORTA bit 1 ON
```

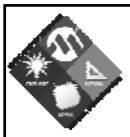

Read-Modify-Write Issue

At Low Frequency or Low Capacitive Loading



At High Frequency or High Capacitive Loading





Solution for RMW Issue

- **Use RMW instruction on the LATch:**
  - BSF LATA,#0
  - BSF LATA,#1

LAB1 Summary

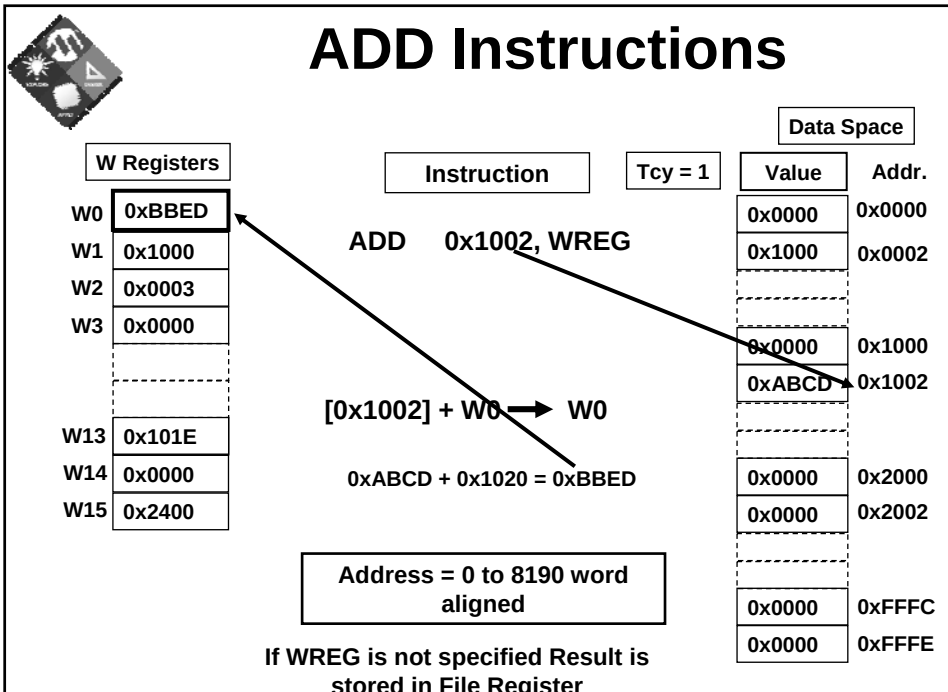
- **Wrote our First 16-bit PIC code ☺**
- **Used some of the Assembly Language Inst.**
- **Learned how to program a 16-bit PIC device using ICD2**
- **Got the Program to WORK!!**

Instruction Set Functional Groups

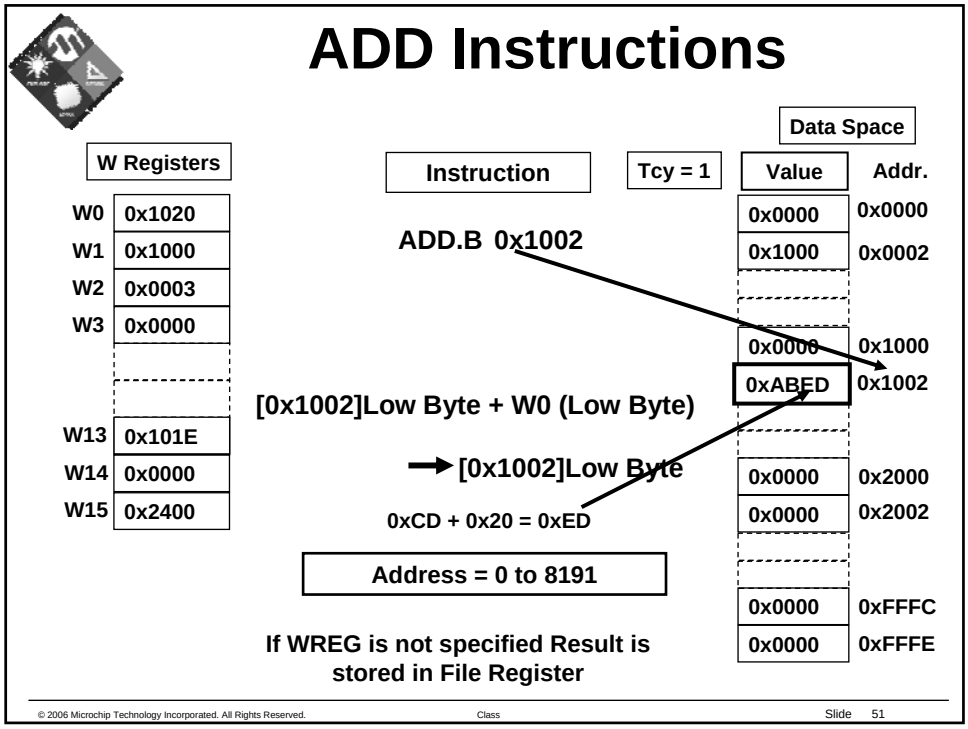
- **16-bit Instructions**
  - **Move Instructions**
  - **Bit Instructions**
    - Follow along LAB 1
  - **Math and Logic Instructions**
  - **Stack Control Instructions**
  - **Program Flow Control Instructions**
  - **CPU Control Instructions**
    - LAB 2

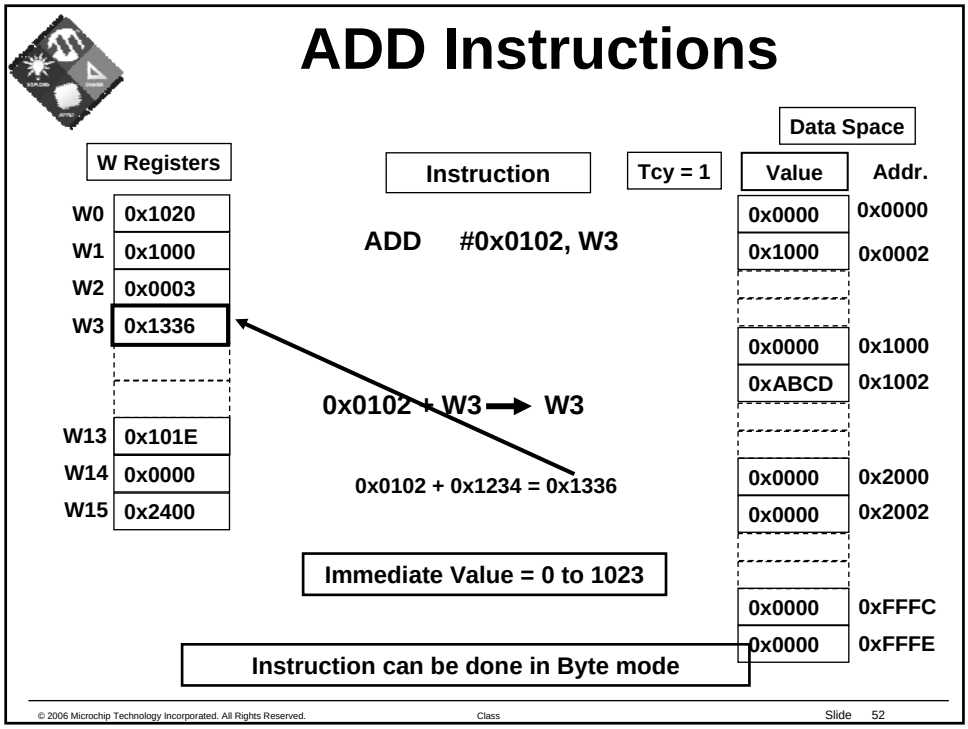
 Class Slide 49

ADD Instructions

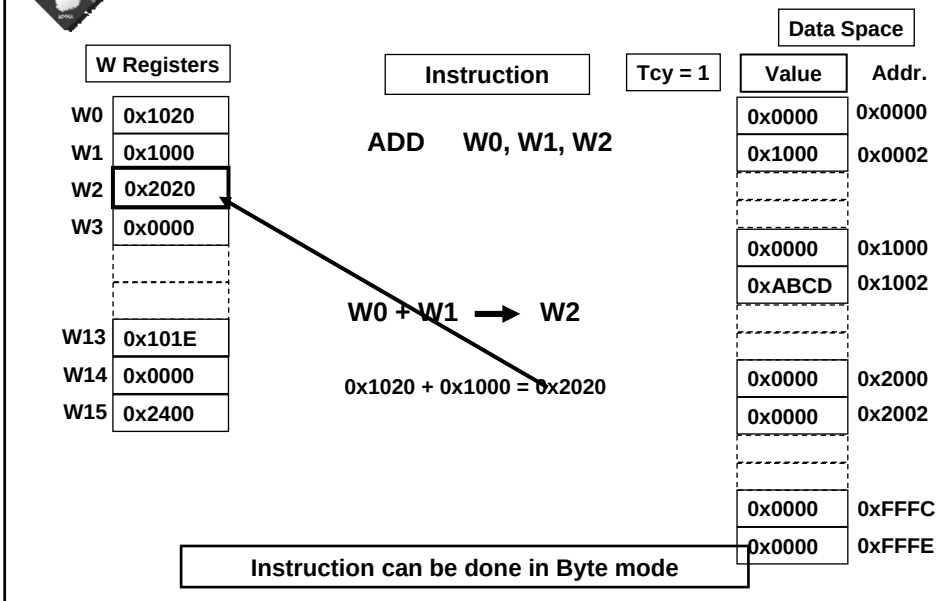


 Class Slide 50


ADD Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x1020
W1 0x1000
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
ADD.B 0x1002
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABED 0x1002
[0x1002]Low Byte + W0 (Low Byte)
[0x1002]Low Byte
0x0000 0x2000
0x0000 0x2002
0xCD + 0x20 = 0xED
Address = 0 to 8191
0x0000 0xFFFC
0x0000 0xFFFE
If WREG is not specified Result is stored in File Register
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 51


ADD Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x1020
W1 0x1000
W2 0x0003
W3 0x1336
W13 0x101E
W14 0x0000
W15 0x2400
ADD #0x0102, W3
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0102 + W3 W3
0x0102 + 0x1234 = 0x1336
0x0000 0x2000
0x0000 0x2002
Immediate Value = 0 to 1023
0x0000 0xFFFC
0x0000 0xFFFE
Instruction can be done in Byte mode
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 52

ADD Instructions

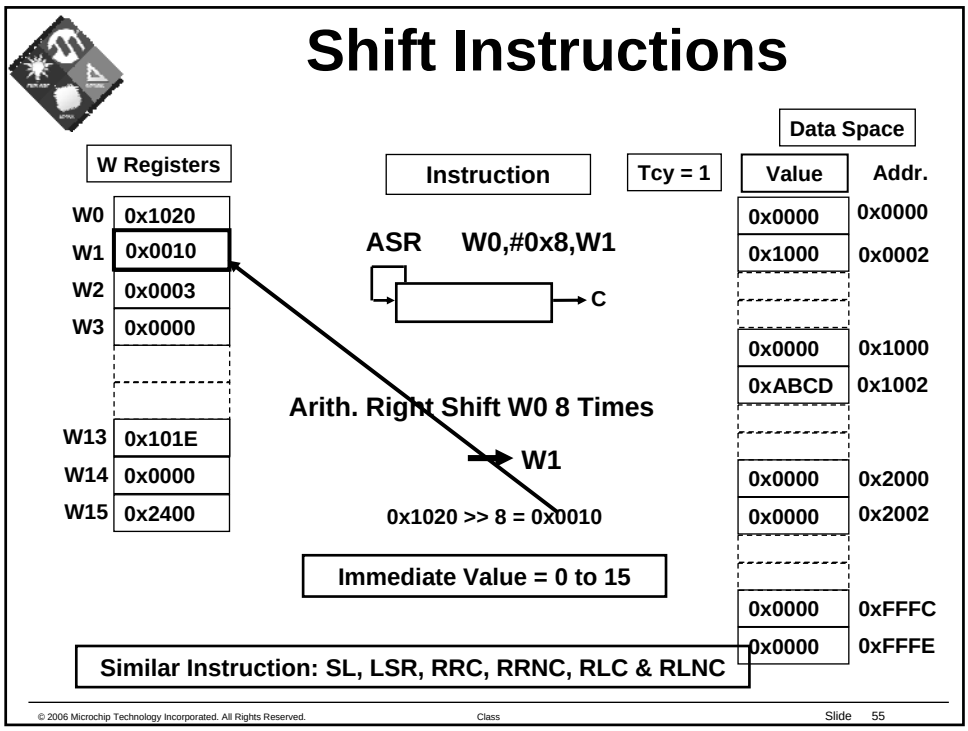


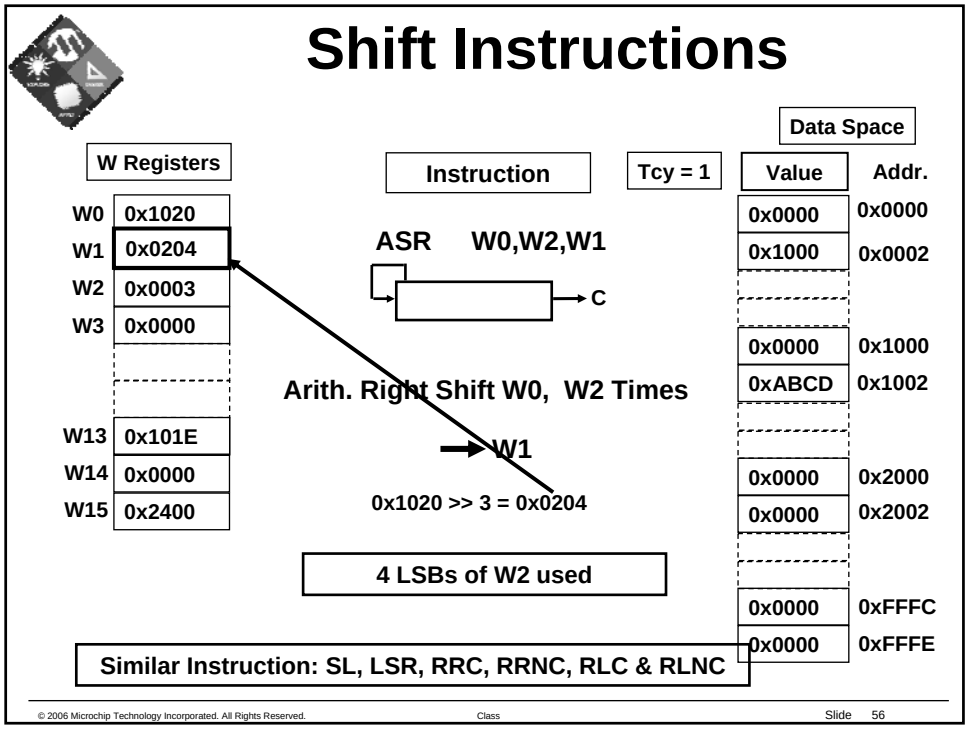
 Class Slide 53

Similar Instruction to ADD

- ADDC, SUB and SUBB
- INC, INC2, DEC and DEC2
- SUBBR and SUBR

 Class Slide 54

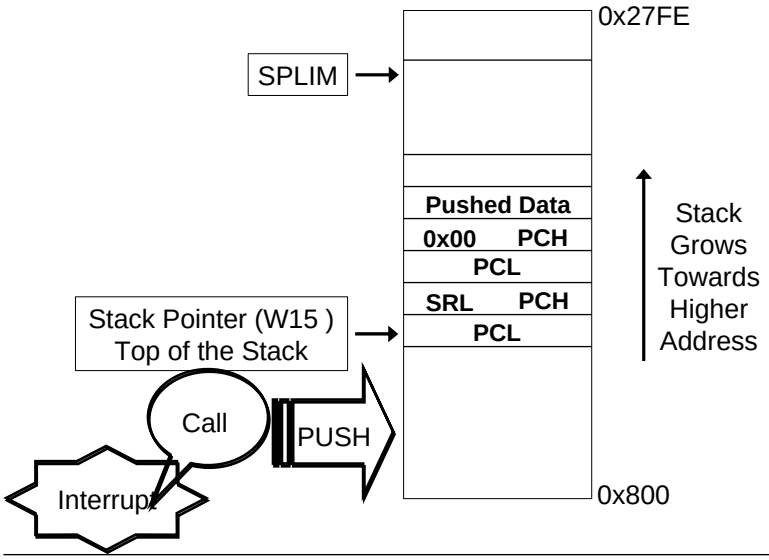

Shift Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x1020
W1 0x0010
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
ASR W0,#0x8,W1
C
Arith. Right Shift W0 8 Times
W1
0x1020 >> 8 = 0x0010
Immediate Value = 0 to 15
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Similar Instruction: SL, LSR, RRC, RRNC, RLC & RLNC
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 55


Shift Instructions
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x1020
W1 0x0204
W2 0x0003
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
ASR W0,W2,W1
C
Arith. Right Shift W0, W2 Times
W1
0x1020 >> 3 = 0x0204
4 LSBs of W2 used
0x0000 0x0000
0x1000 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Similar Instruction: SL, LSR, RRC, RRNC, RLC & RLNC
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 56

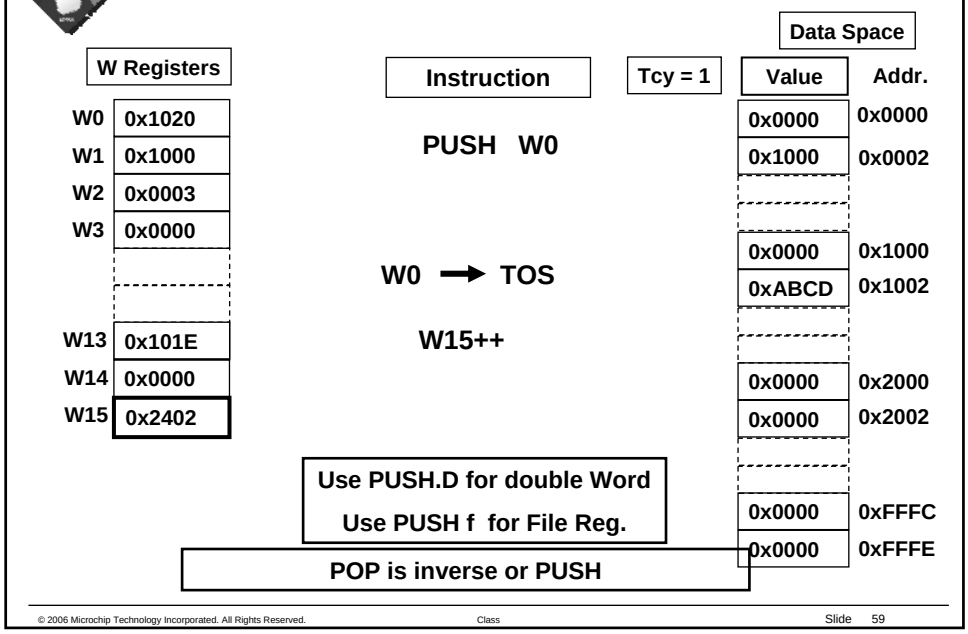
Instruction Set Functional Groups

- **16-bit Instructions**
  - **Move Instructions**
  - **Bit Instructions**
    - Follow along LAB 1
  - **Math and Logic Instructions**
  - **Stack Control Instructions**
  - **Program Flow Control Instructions**
  - **CPU Control Instructions**
    - LAB 2

Software Stack in Data RAM



PUSH Instructions



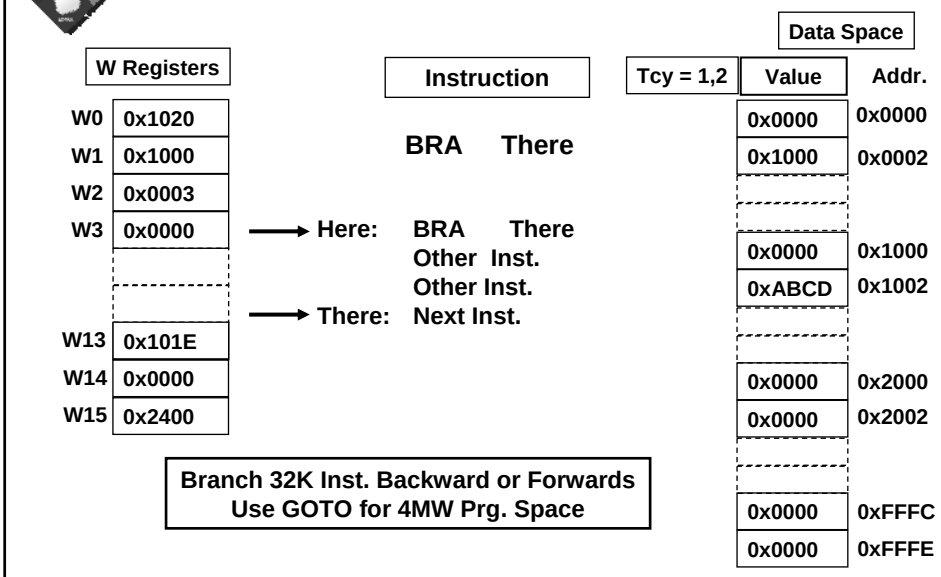
 Class Slide 59

Instruction Set Functional Groups

- 16-bit Instructions
  - Move Instructions
  - Bit Instructions
    - Follow along LAB 1
  - Math and Logic Instructions
  - Stack Control Instructions
  - Program Flow Control Instructions
  - CPU Control Instructions
    - LAB 2

 Class Slide 60

Program Flow Instructions



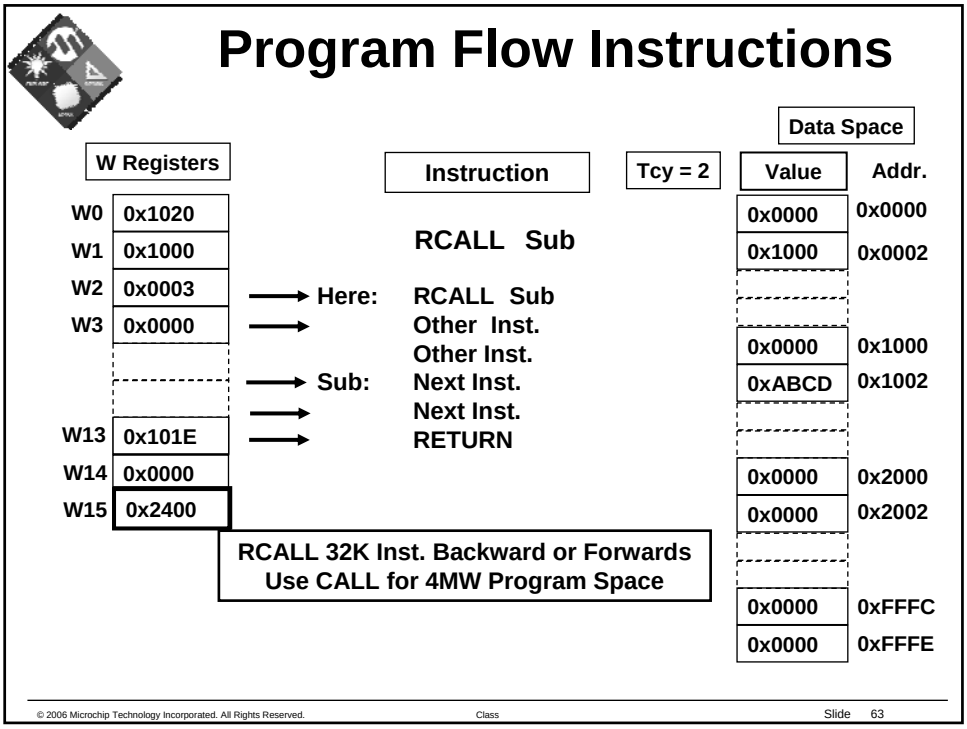
 Class Slide 61

Conditional Branches

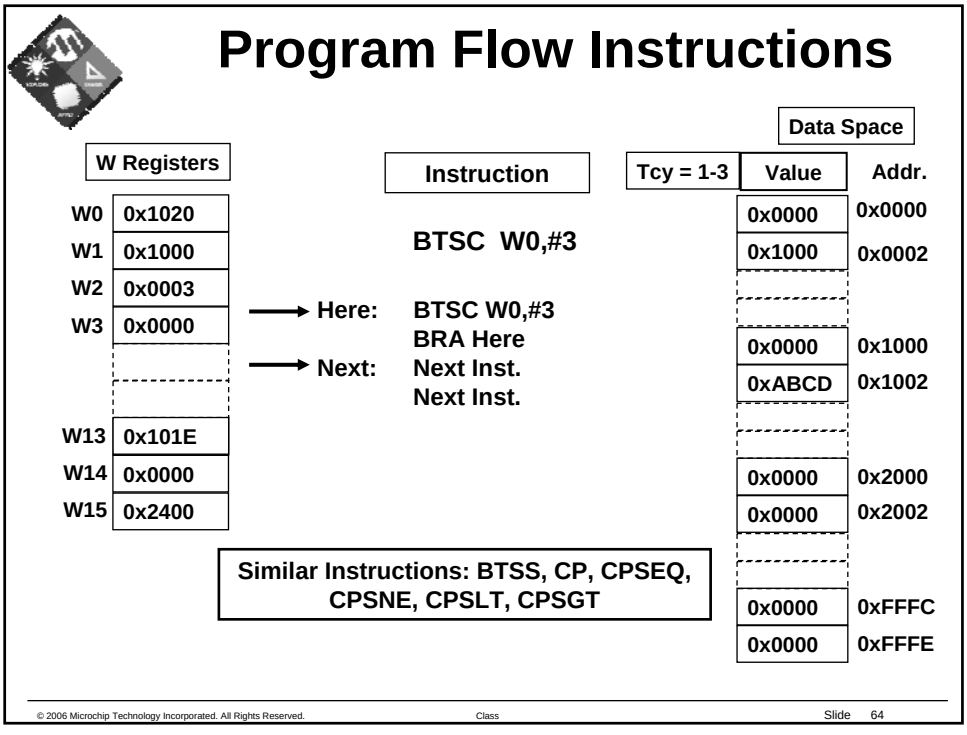
- BRA C There ; if Carry Set
- BRA NC There ; if Not Carry
- BRA GT There ; if Signed Greater Than
- BRA GTU There ; if UnSigned Greater Than
- BRA LT There ; if Signed Less Than
- BRA LTU There ; if Unsigned Less Than
- BRA Z There ; if Zero
- BRA NZ There ; if Not Zero
- BRA N There ; if Negative
- BRA NN There ; if Not Negative

 Class Slide 62

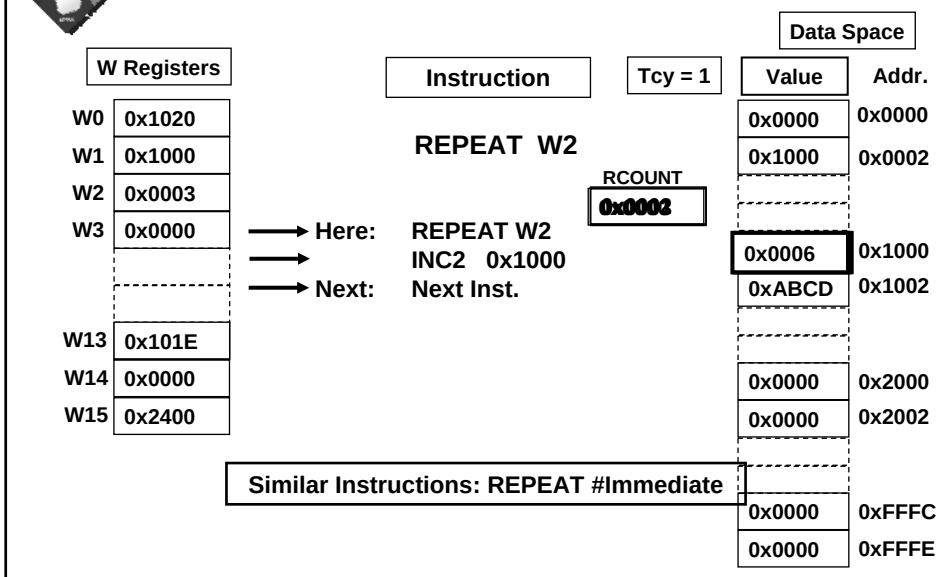
Program Flow Instructions



Program Flow Instructions



Program Flow Instructions



 Class Slide 65

Instruction Set Functional Groups

- **16-bit Instructions**
  - **Move Instructions**
  - **Bit Instructions**
    - Follow along LAB 1
  - **Math and Logic Instructions**
  - **Stack Control Instructions**
  - **Program Flow Control Instructions**
  - **CPU Control Instructions**
    - LAB 2

 Class Slide 66

Control Operations

- **CLRWDT –**
  - Clears the Watch Dog Timer
- **DISI #Lit14 –**
  - Disables Interrupts for 14-bit Literal value *Tcyc
- **PWRSAV**
  - Use to put CPU in SLEEP or IDLE States
- **RESET**
  - Software Device Reset
- **NOP**

 Class Slide 67

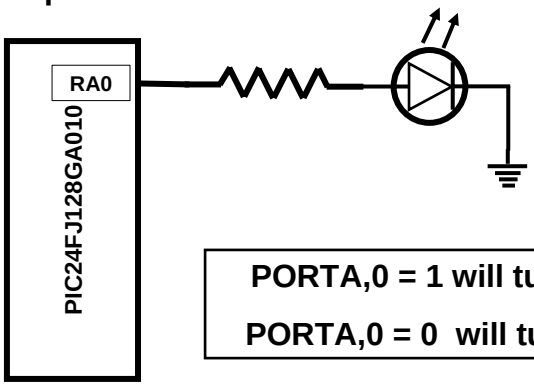
Session Agenda

- 16-bit Architecture Basics
- Some 16-bit Assembly Instructions
- I/O Port Handling
  - **Hands-on Lab #1 Follow along**
    - **Basic Assembly language setup**
    - **Light LED on I/O Port**
- Complete 16-bit Assembly Instructions
  - **Hands-on Lab #2**
    - **Use Loop Instructions to Blink LED**
- Addressing Modes
- Interrupts and Timers
  - **Hands-on Lab #3**
    - **Use Interrupt to Blink LED**

 Class Slide 68

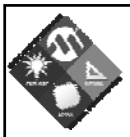
Lab 2: The Task

- Turn On LED at RA0
- Wait for 0.5 seconds using Software Delay
- Turn OFF LED at RA0
- Wait for 0.5 Seconds using Software Delay
- Repeat



PORTA,0 = 1 will turn on LED

PORTA,0 = 0 will turn off LED

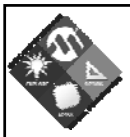


LAB2

- **Open Lab2.mcw workspace in MPLAB**
  - C:\rtc\103_ASP\Lab2.mcw
- **Edit code:**
  - Write a Software Loop Delay of about 0.5 Second
  - Turn ON LED
  - Wait 0.5 Second
  - Turn OFF LED, delay half second and Repeat
- **Compile Code**
- **Program Part on Explorer-16 using ICD2**
- **Run Code using ICD2**
- **See the LED BLINK!**

LAB 2 Clock Speed Hint

- **Use Primary Oscillator Mode, HS Osc**
- **XT Xtal = 8.00 Mhz = Fosc**
- **FCY = 4000000 (Fosc/2)**
- **Instruction Speed = 4 Mhz**
- **So Tcy = 250 nS**
- **0.5 Sec Software Delay = 2Million Instructions**
- **Use w2 register as 16-bit counter**
  - 30 Tcy * 65536(16-bit) = 1.967 Million Instructions
  - Use Repeat Instruction to get the 30 Instructions



LAB2 Summary

- **Wrote some more 16-bit PIC code**
- **Used some more Assembly Language Inst.**
  - Branch
  - Repeat Instruction
  - Loop Instructions
- **Got the Program to WORK!!**

 Class Slide 73

Session Agenda

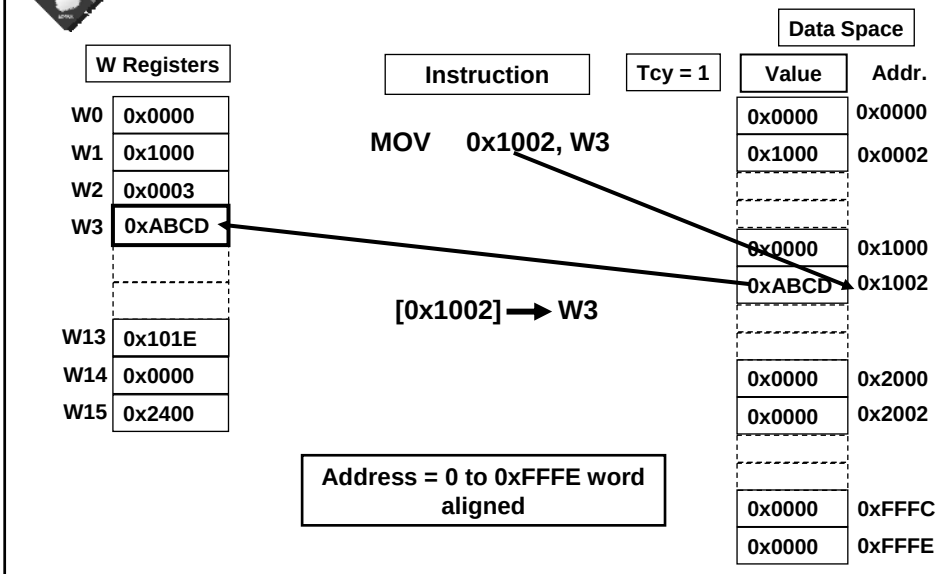
- 16-bit Architecture Basics
- Some 16-bit Assembly Instructions
- I/O Port Handling
  - **Hands-on Lab #1 Follow along**
    - **Basic Assembly language setup**
    - **Light LED on I/O Port**
- Complete 16-bit Assembly Instructions
  - **Hands-on Lab #2**
    - **Use Loop Instructions to Blink LED**
- Addressing Modes
- Interrupts and Timer1
  - **Hands-on Lab #3**
    - **Use Interrupt to Blink LED**

 Class Slide 74

Basic Addressing Modes

- File Register / Memory Direct
  - Register Direct
  - Register Indirect with optional pre- or post-modification
  - Immediate
  - Register indirect with register or literal offset

MOV Instructions



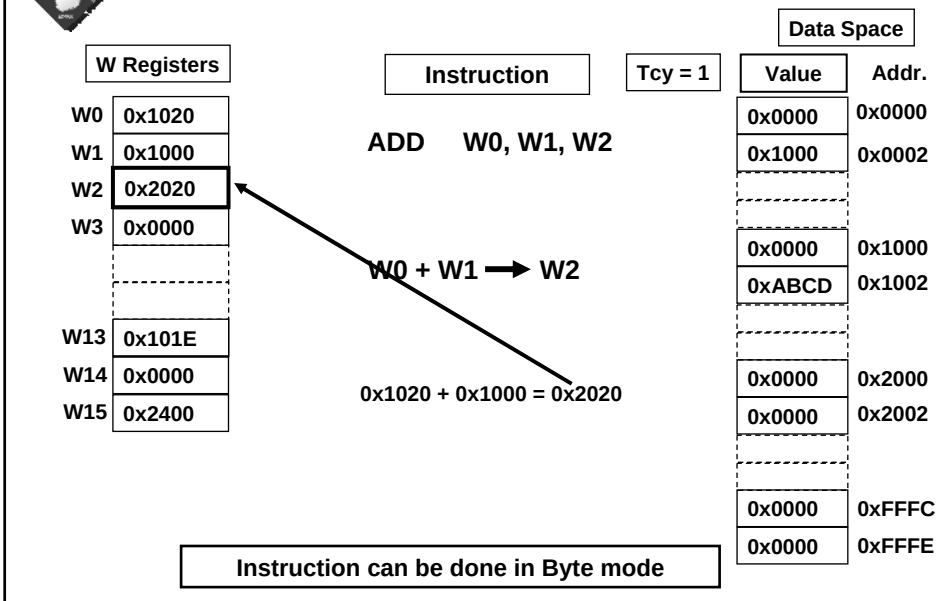
 Class Slide 77

Basic Addressing Modes

- **Basic addressing modes:**
  - File Register / Memory Direct
  - → Register Direct
  - Register Indirect with optional pre- or post-modification
  - Immediate
  - Register indirect with register or literal offset

 Class Slide 78

Register Direct

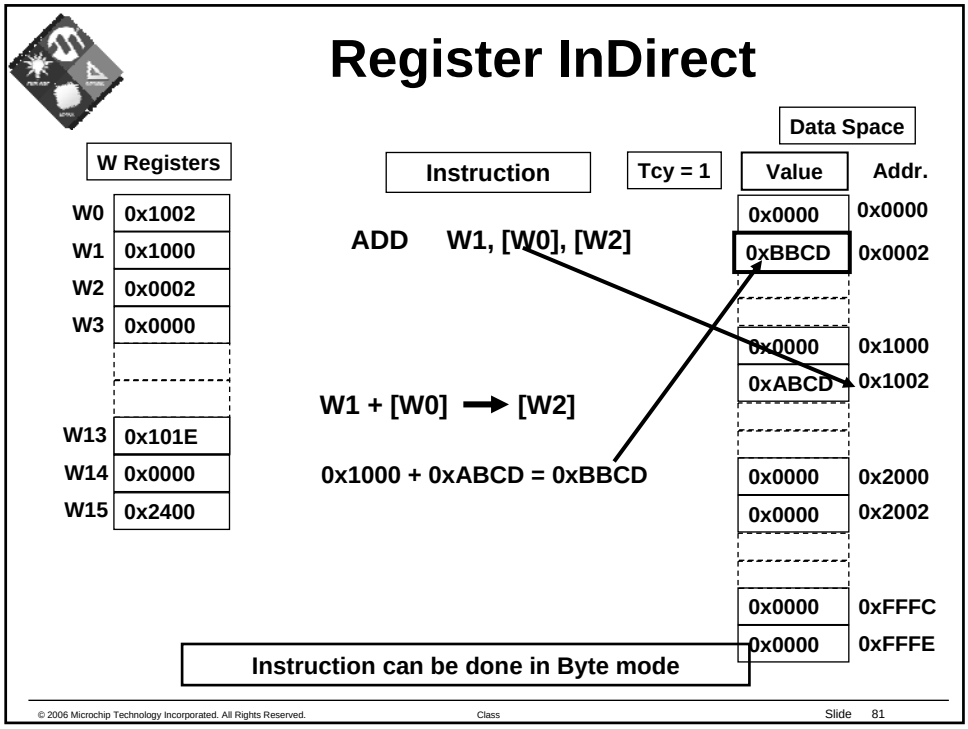


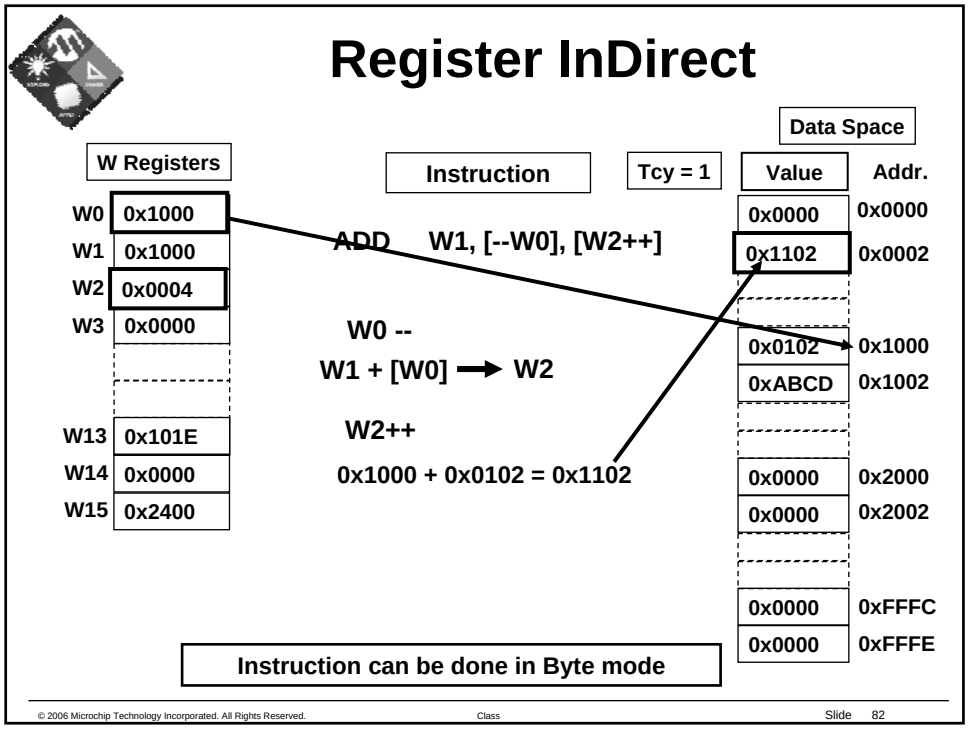
 Class Slide 79

Basic Addressing Modes

- **Basic addressing modes:**
  - File Register / Memory Direct
  - Register Direct
  - → Register Indirect with optional pre- or post-modification
  - Immediate
  - Register indirect with register or literal offset

 Class Slide 80


Register InDirect
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x1002
W1 0x1000
W2 0x0002
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
ADD W1, [W0], [W2]
W1 + [W0] → [W2]
0x1000 + 0xABCD = 0xBBCD
0x0000 0x0000
0xBBCD 0x0002
0x0000 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Instruction can be done in Byte mode
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 81


Register InDirect
Data Space
W Registers
Instruction
Tcy = 1
Value
Addr.
W0 0x1000
W1 0x1000
W2 0x0004
W3 0x0000
W13 0x101E
W14 0x0000
W15 0x2400
ADD W1, [--W0], [W2++]
W0 --
W1 + [W0] → W2
W2++
0x1000 + 0x0102 = 0x1102
0x0000 0x0000
0x1102 0x0002
0x0102 0x1000
0xABCD 0x1002
0x0000 0x2000
0x0000 0x2002
0x0000 0xFFFC
0x0000 0xFFFE
Instruction can be done in Byte mode
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 82

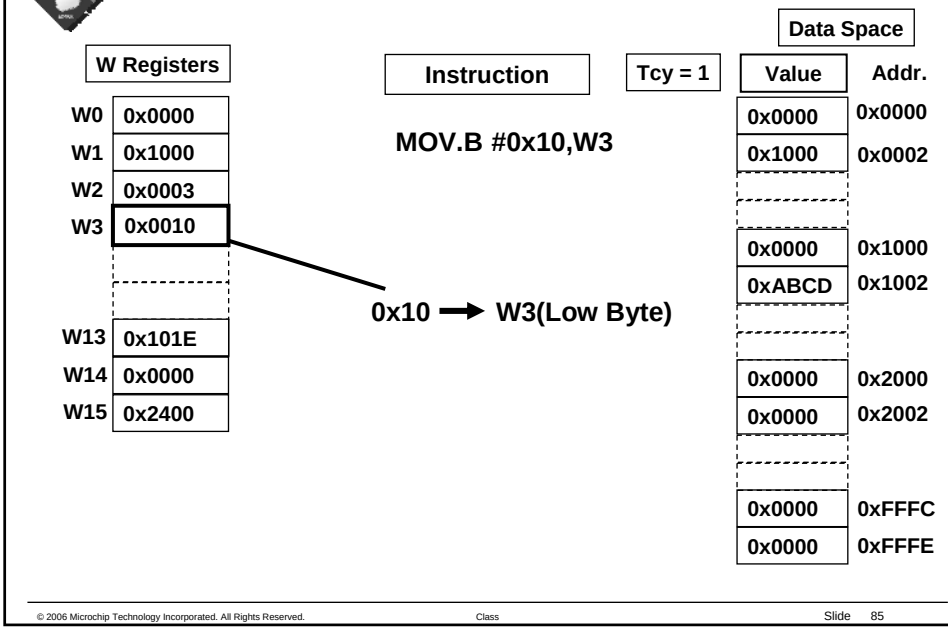
Register Indirect Examples

- **With pre/post increment**
  - MOV [++W4], [W6++]
- **With pre/post decrement**
  - MOV [W4++], [W6--]
- **With register offset**
  - MOV [W4 + W5], [W6]
- **With literal offset**
  - MOV [W4 + #768]. [W6]

Basic Addressing Modes

- **Basic addressing modes:**
  - File Register / Memory Direct
  - Register Direct
  - Register Indirect with optional pre- or post-modification
  - → Immediate
  - Register indirect with register or literal offset

MOV Instructions



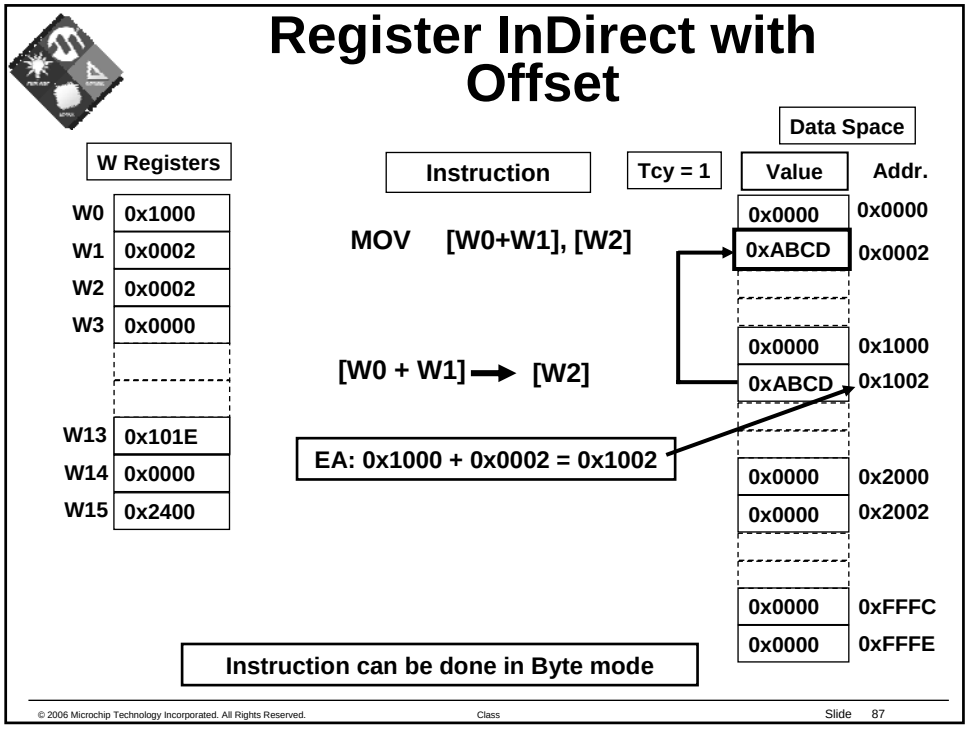
 Class Slide 85

Basic Addressing Modes

- **Basic addressing modes:**
  - File Register / Memory Direct
  - Register Direct
  - Register Indirect with optional pre- or post-modification
  - Immediate
  - Register indirect with register or literal offset

 Class Slide 86

Register InDirect with Offset

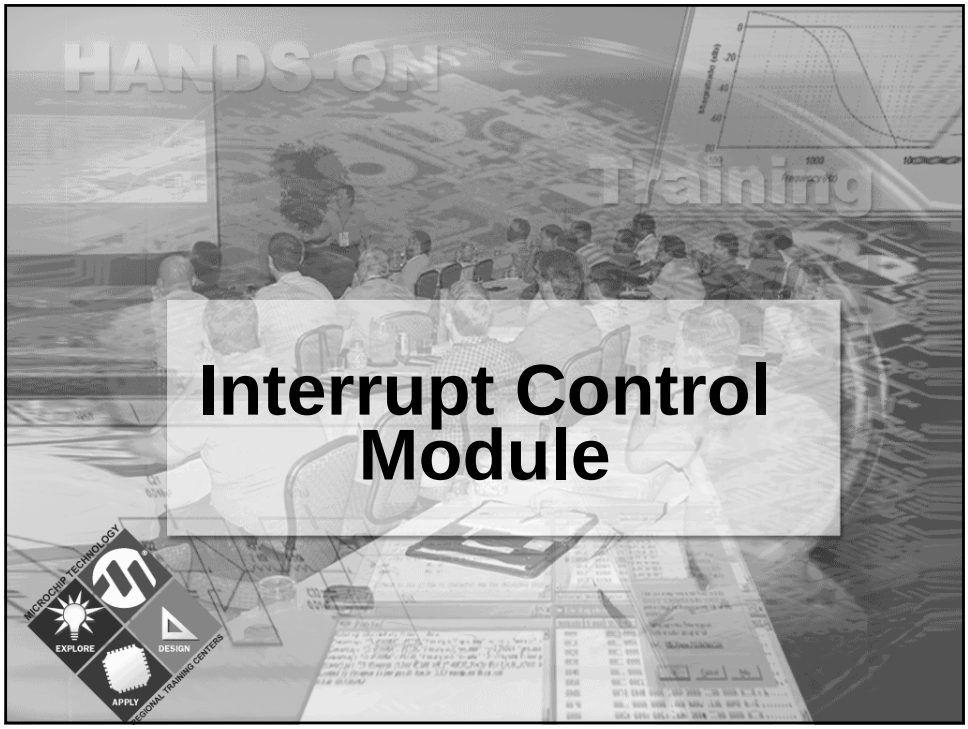


 Class Slide 87

Session Agenda

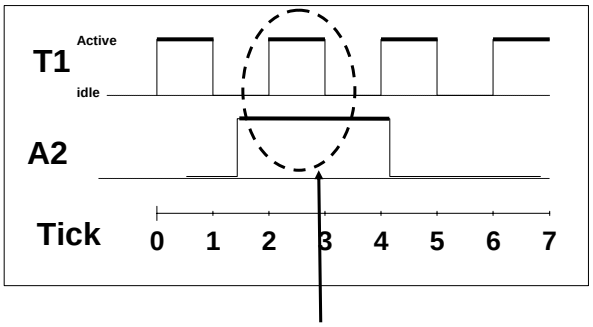
- 16-bit Architecture Basics
- Some 16-bit Assembly Instructions
- I/O Port Handling
  - **Hands-on Lab #1 Follow along**
    - **Basic Assembly language setup**
    - **Light LED on I/O Port**
- Complete 16-bit Assembly Instructions
  - **Hands-on Lab #2**
    - **Use Loop Instructions to Blink LED**
- Addressing Modes
- Interrupts and Timer1
  - **Hands-on Lab #3**
    - **Use Interrupt to Blink LED**

 Class Slide 88



Interrupt Driven System

```
main
{
  while(1);
}

T1_ISR
{

}

A2_ISR
{

}
```



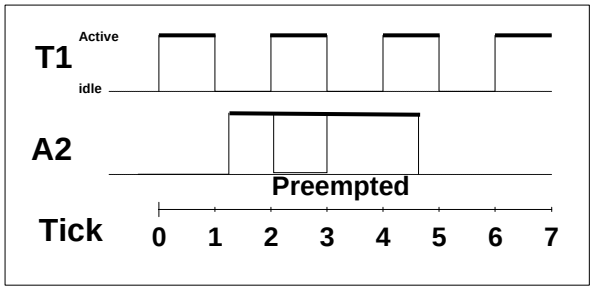
Two Interrupts are active

	Interrupt Rate	Interrupt Execution Time
Timer1	50us (20khz)	25uS
A2	Async	65uS

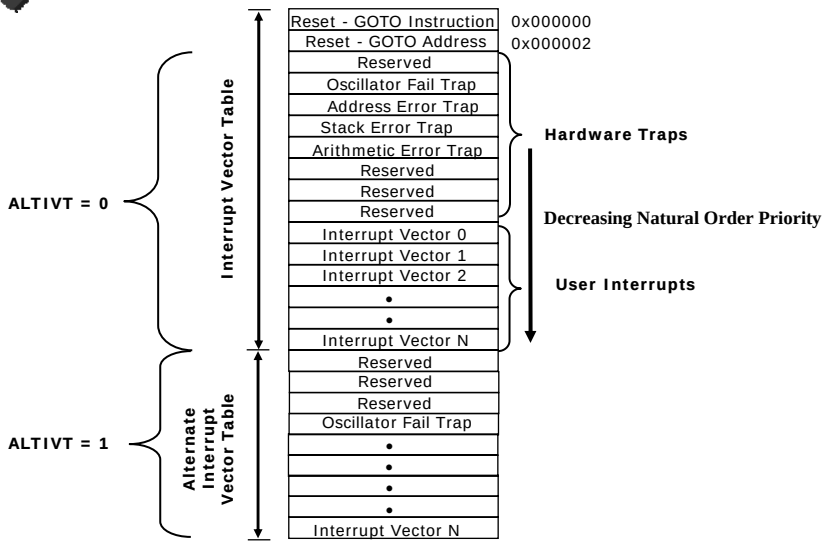
Interrupt Nesting

- In 16-bit Arch T1 assigned a higher Priority than A2
- When T1 occurs during A2, T1 preempts A2

```
main
{
 while(1);
}

T1_ISR
{

}

A2_ISR
{

}
```



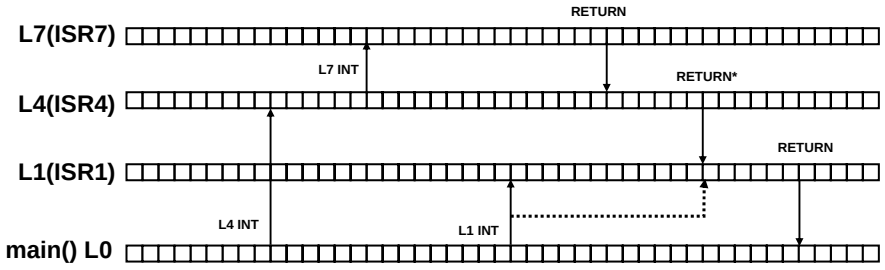
Interrupt Vector Table

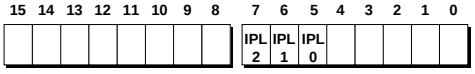


Interrupt Priority

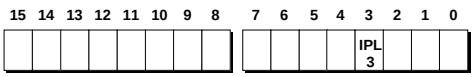
- CPU has 16 priority levels
  - **Level 0 is the default CPU level (**main**)**
  - **Level 1 - 7 for user interrupts**
  - **Level 8 -15 reserved for traps (**NMI**)**
- Interrupt with priority level greater than current CPU level (IPL<3:0>) can interrupt the CPU
- Interrupts are nested by default, Nesting can be disabled by setting NSTDIS bit in INTCON1 register
- IVT has natural priority to resolve conflicts
- User assigned priority overrides natural priority

Interrupt Nesting Example





Status Register



Core Control Register

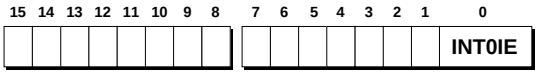
Disabling Interrupts

- **DISI Instruction**
  - DISI #N instruction disables level 1 - 6 interrupts for (N+1) Instruction cycle
- **Write to CPU** IPL<2:0> **bits to raise CPU priority to level 7**
  - MOV.B #0xE0, w0
  - MOV.B WREG, SRL

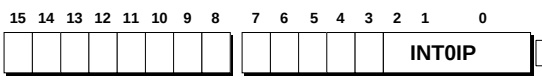
 Class Slide 95

Configuring Interrupts: Example using INT0

IEC0: Interrupt Enable Control Register 0 : Enable Interrupt

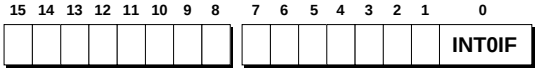


IPC0: Interrupt Priority Control Register 0 : Assign Priority



000: Disabled
001: Priority 1 Interrupt
010: Priority 2 Interrupt
011: Priority 3 Interrupt
100: Priority 4 Interrupt
101: Priority 5 Interrupt
110: Priority 6 Interrupt
111: Priority 7 Interrupt

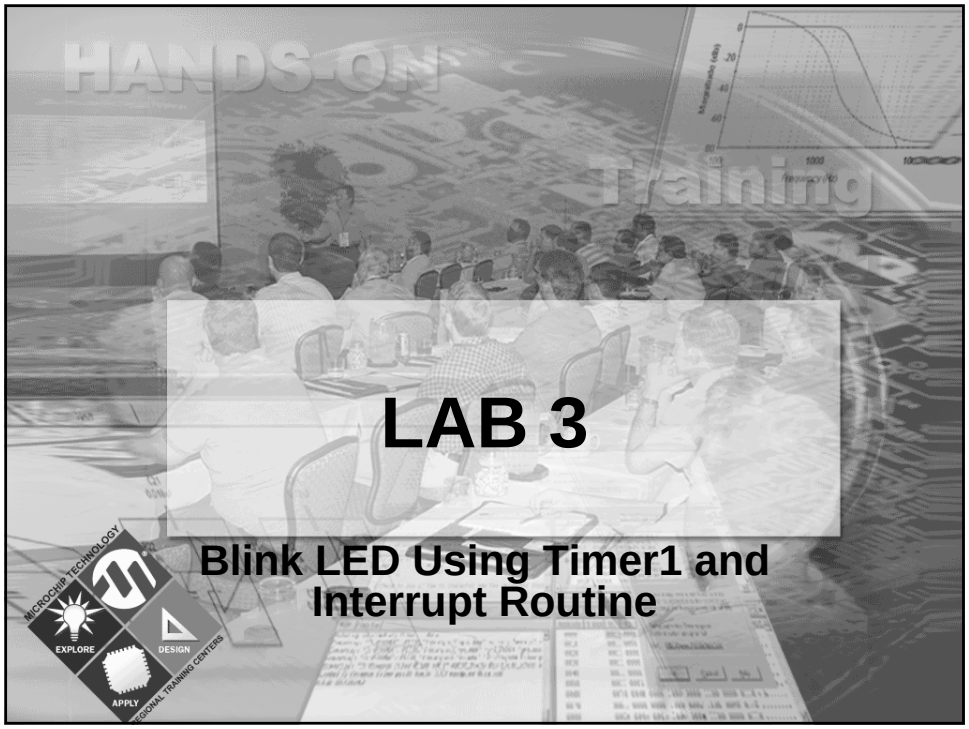
IFS0: Interrupt Flag Control Register 0 : Clear Interrupt in the ISR



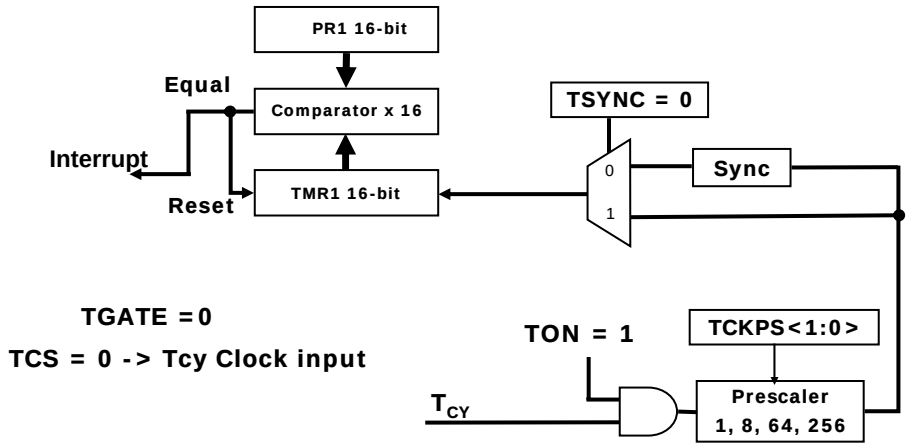
 Class Slide 96

Traps for Robust Operation

- **Oscillator Failure Trap (level 14)**
- **Address Error Trap (level 13)**
  - Instruction fetch from illegal program space
  - Data fetch from unimplemented data space
  - Unaligned word access from data space
- **Stack Error Trap (level 12)**
  - Stack overflow or underflow
- **Math Error Trap (level 11)**
  - Divide by Zero
  - Unsaturated Accumulator Overflow (A or B)
  - Catastrophic Accumulator Overflow (either)
  - Accumulator Shift Overflow

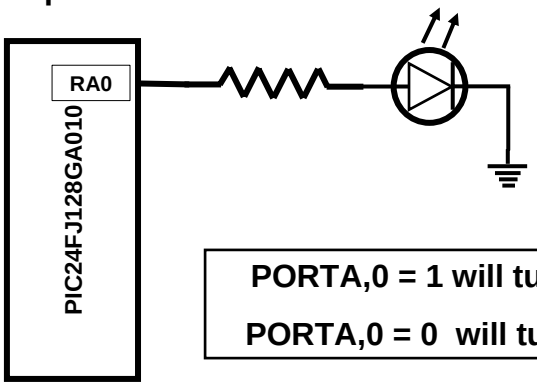


Simplified TMR1 Diagram



Lab 3: The Task

- Turn On LED at RA0
- Wait for 0.5 seconds using TMR1 Delay
- Turn OFF LED at RA0
- Wait for 0.5 Seconds using TMR1 Delay
- Repeat



PORTA,0 = 1 will turn on LED

PORTA,0 = 0 will turn off LED

Lab3 Clock Hints

- **Clock TMR1 using internal Tcy**
- **Fcy = 4Mhz**
- **Divide Fcy by 64: Set T1CON,#5**
- **For Half Second Interrupts**
  - PR1 value = (Fcy/64)* 0.5
- **Turn on TMR1: Set T1CON,#TON**
- **Interrupt Service Routine Provided:**
  - Clear the interrupt Flag: T1IF in IFS0 register
  - Remember code gets here ever 0.5 Secs
  - Write code to blink the LED
- **After initialization, do nothing in main program**

 Class Slide 101

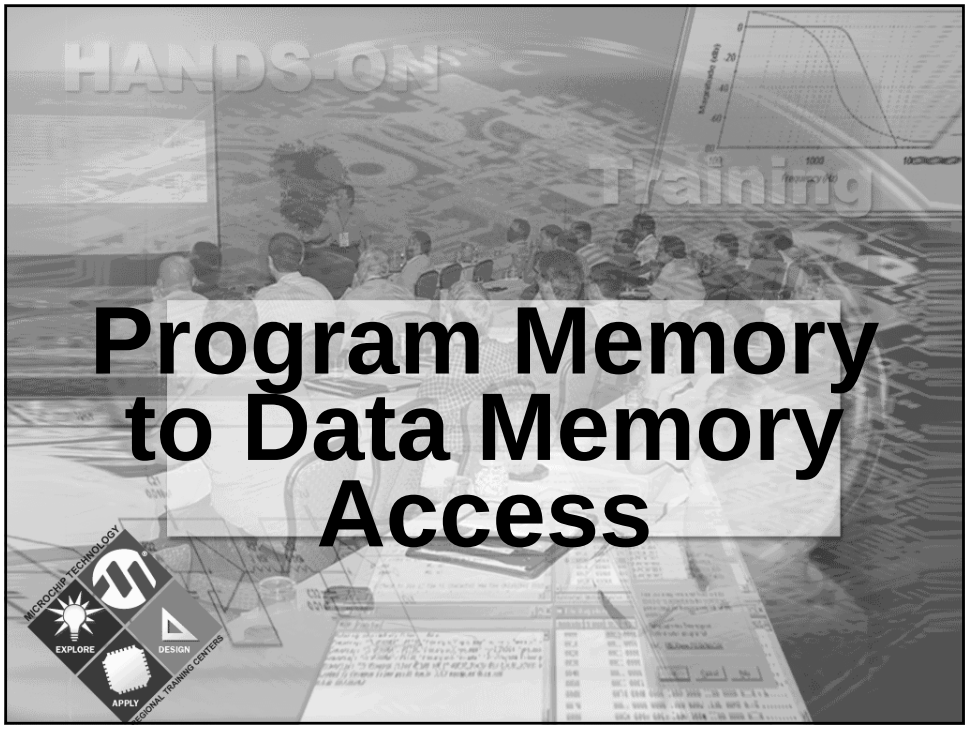
LAB3

- **Open Lab3.mcw workspace in MPLAB**
  - C:\rtc\103_ASP\Lab3.mcw
- **Edit code:**
  - Initialize Timer 1 for a delay of 500 mS
  - In Timer1 Interrupt Service Routine toggle LED
- **Compile Code**
- **Program Part on Explorer-16 using ICD2**
- **Run Code using ICD2**
- **See the LED BLINK!**

 Class Slide 102

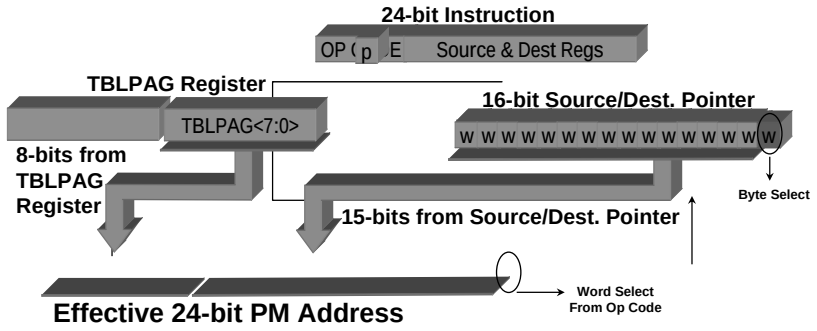
LAB3 Summary

- Learned about Timer1 Peripheral
- Wrote code to Initialize Timer1 Peripheral
- Wrote an Interrupt Service Routine
- Got the Program to WORK!!

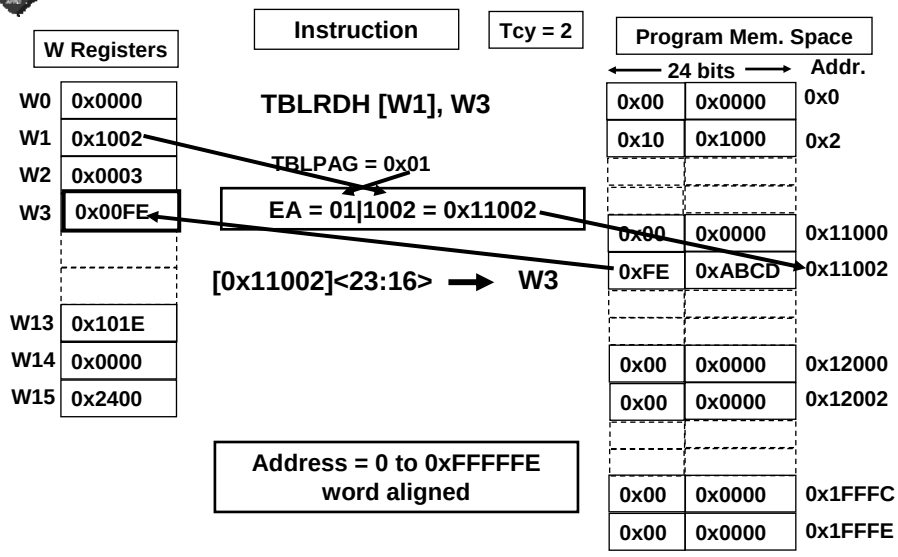


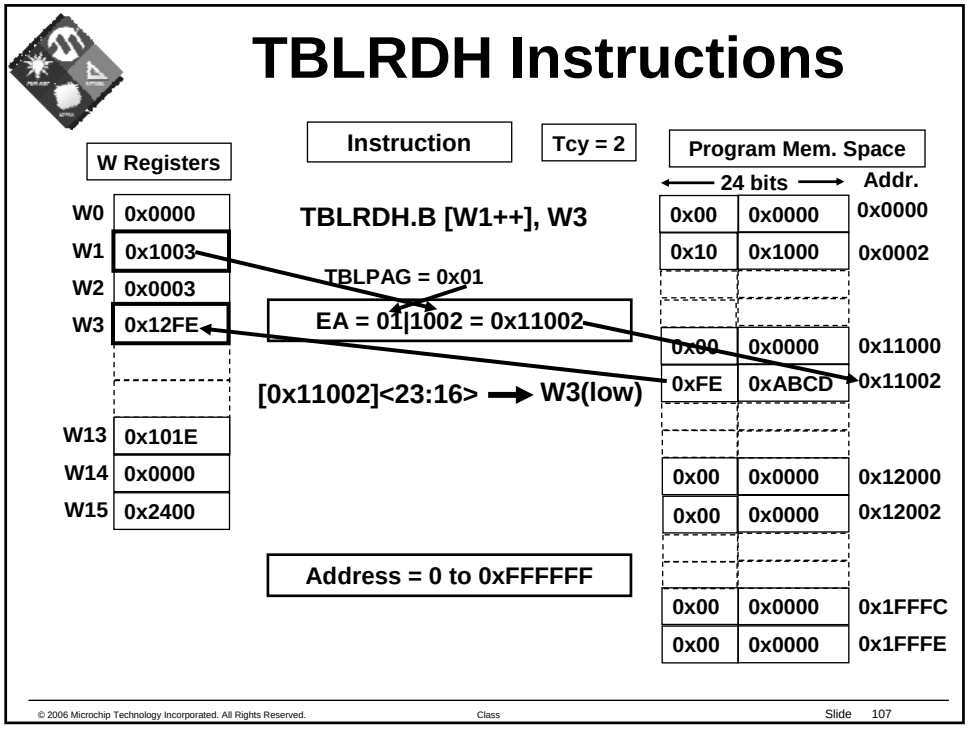
Data Access from PM Using Table Instructions

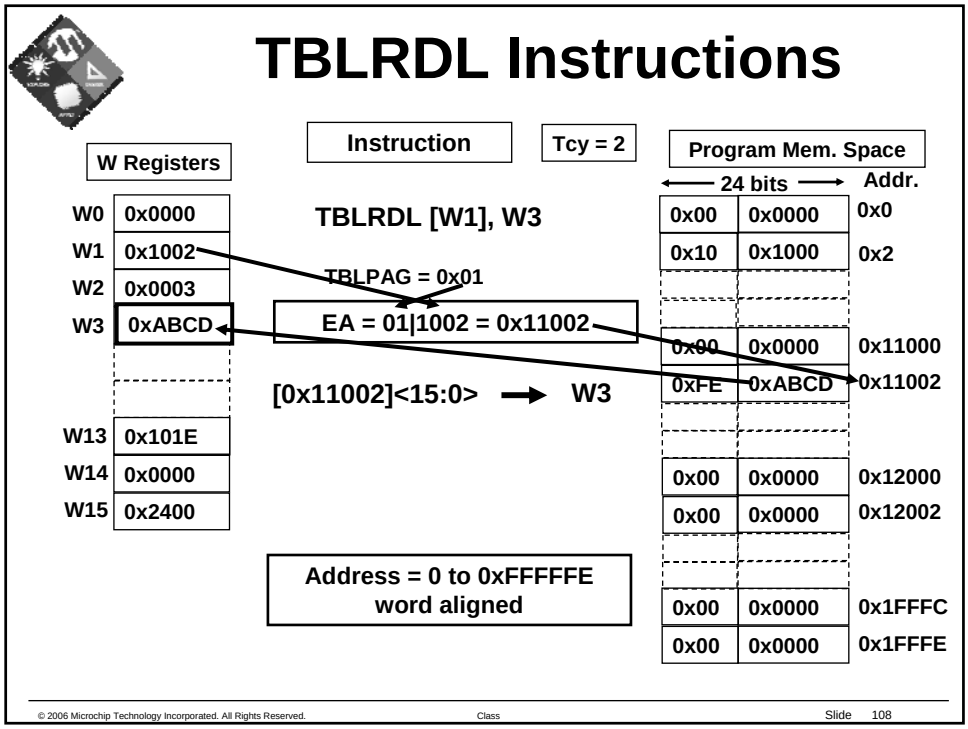
- **24-bit Effective Address(EA) formation**
  - Configuration space is accessed by setting TBLPAG<7> (i.e. EA<23>)
    - **Only means of accessing configuration space**



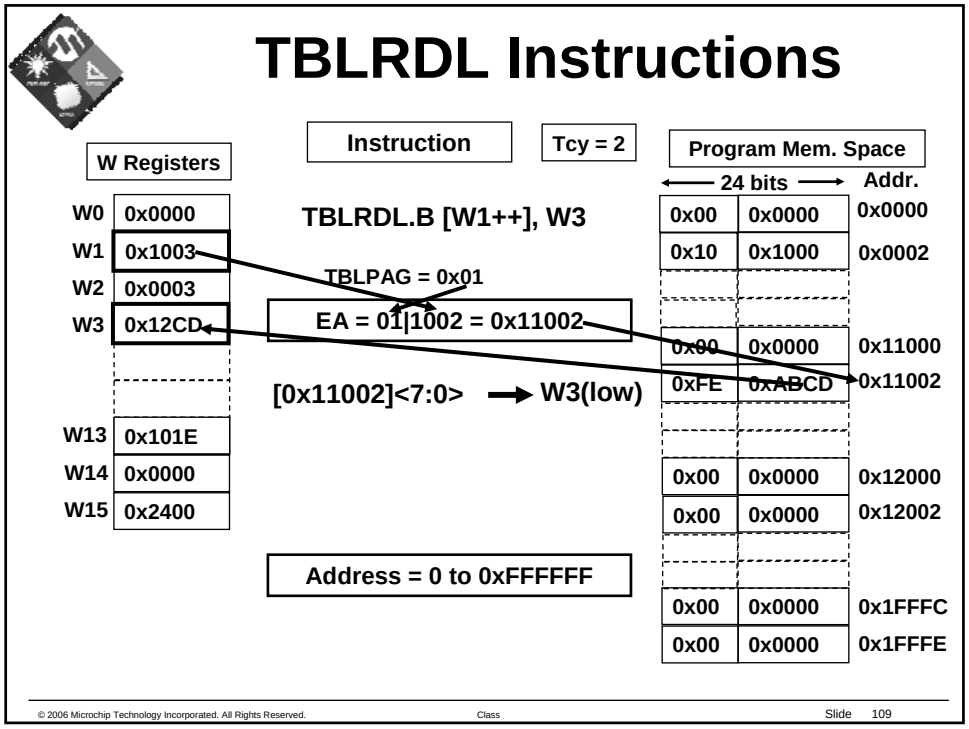
TBLRDH Instructions




TBLRDH Instructions
Instruction
Tcy = 2
W Registers
Program Mem. Space
24 bits
Addr.
W0 0x0000
W1 0x1003
W2 0x0003
W3 0x12FE
W13 0x101E
W14 0x0000
W15 0x2400
TBLRDH.B [W1++], W3
TBLPAG = 0x01
EA = 01|1002 = 0x11002
[0x11002]<23:16> W3(low)
Address = 0 to 0xFFFFFF
0x00 0x0000 0x0000
0x10 0x1000 0x0002
0x00 0x0000 0x11000
0xFE 0xABCD 0x11002
0x00 0x0000 0x12000
0x00 0x0000 0x12002
0x00 0x0000 0x1FFFC
0x00 0x0000 0x1FFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 107


TBLRDL Instructions
Instruction
Tcy = 2
W Registers
Program Mem. Space
24 bits
Addr.
W0 0x0000
W1 0x1002
W2 0x0003
W3 0xABCD
W13 0x101E
W14 0x0000
W15 0x2400
TBLRDL [W1], W3
TBLPAG = 0x01
EA = 01|1002 = 0x11002
[0x11002]<15:0> W3
Address = 0 to 0xFFFFFE
word aligned
0x00 0x0000 0x0
0x10 0x1000 0x2
0x00 0x0000 0x11000
0xFE 0xABCD 0x11002
0x00 0x0000 0x12000
0x00 0x0000 0x12002
0x00 0x0000 0x1FFFC
0x00 0x0000 0x1FFFE
© 2006 Microchip Technology Incorporated. All Rights Reserved.
Class
Slide 108

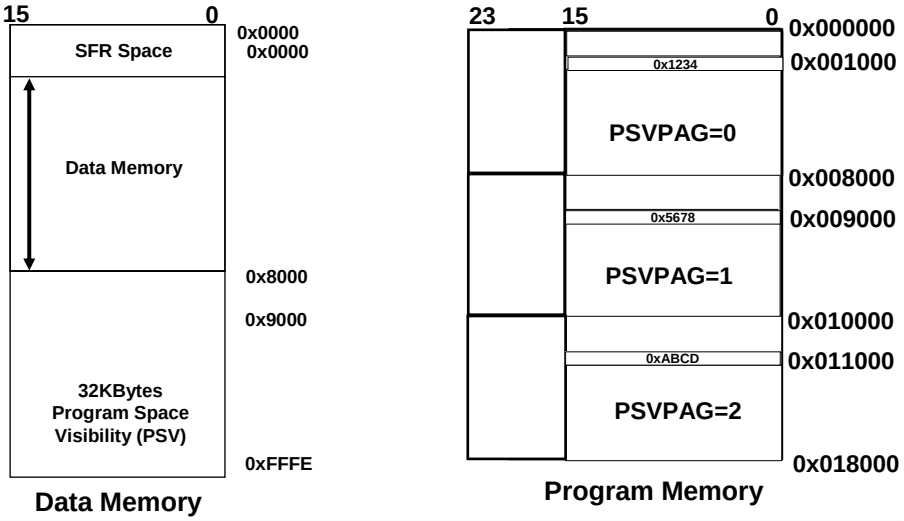
TBLRDL Instructions



 Class Slide 109

Program Space Visibility

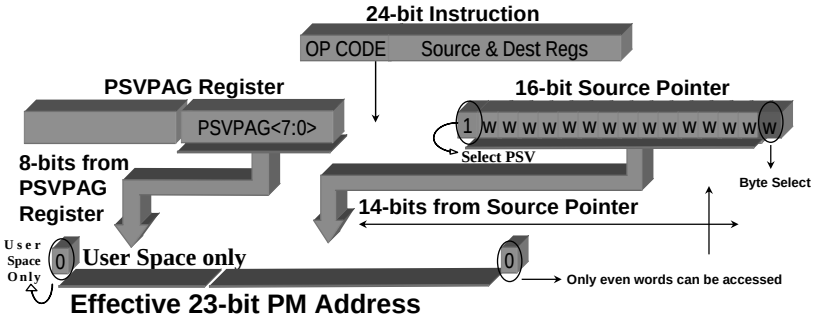
- Program Space Visibility Enabled when CORCON<PSV> = 1
- PSVPAG register maps 32Kb program memory segment into data memory



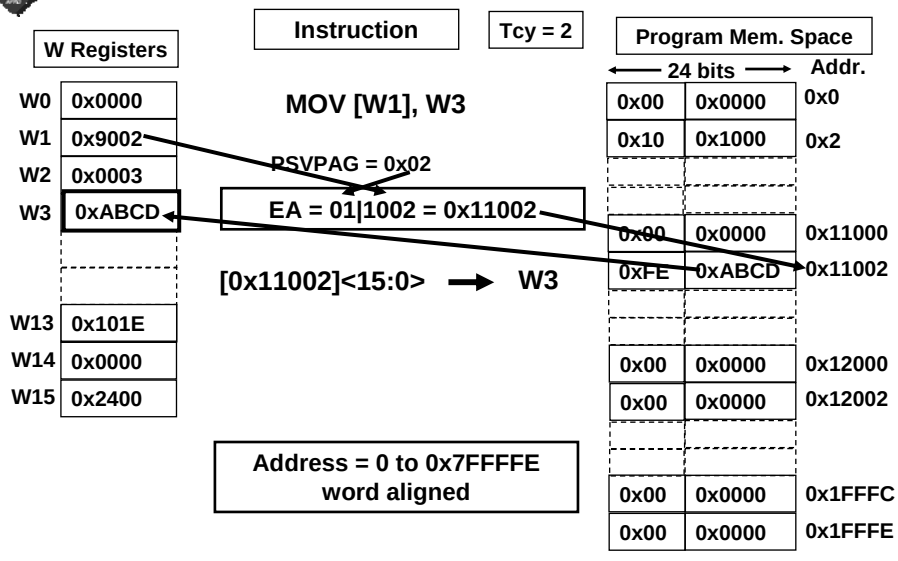
 Class Slide 110

Data Access from PM Using Program Space Visibility

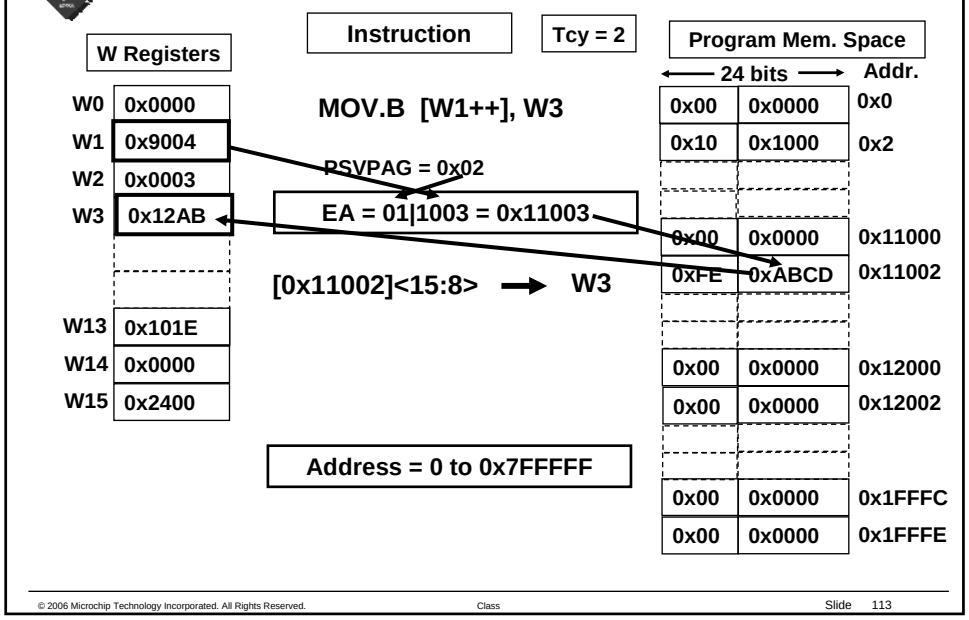
- **32 KB segment of PM may be mapped into the data memory address space**
  - If PSV bit (CORCON<2>) is set and if SrcPointer (DM EA)<15> is '1' then data is accessed from PM



MOV Instructions (PSV)



MOV Instructions (PSV)



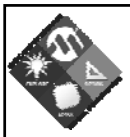
Defining/Accessing PSV Constants

- **Defining constants in PM for PSV, in assembly**

```
    .section .const, psv
hello:
    .ascii “Hello World!\n”
```

- **Accessing PSV in assembly**

```
mov       #psvpage(hello),w0
mov       wo, PSVPAG    ; set PSVPAG address
bset.b    CORCONL,#PSV  ; enable PSV operation
mov       #psvoffset(hello),w0
mov.b     [w0++],w1     ;get first byte
```

Usefulness of PSV

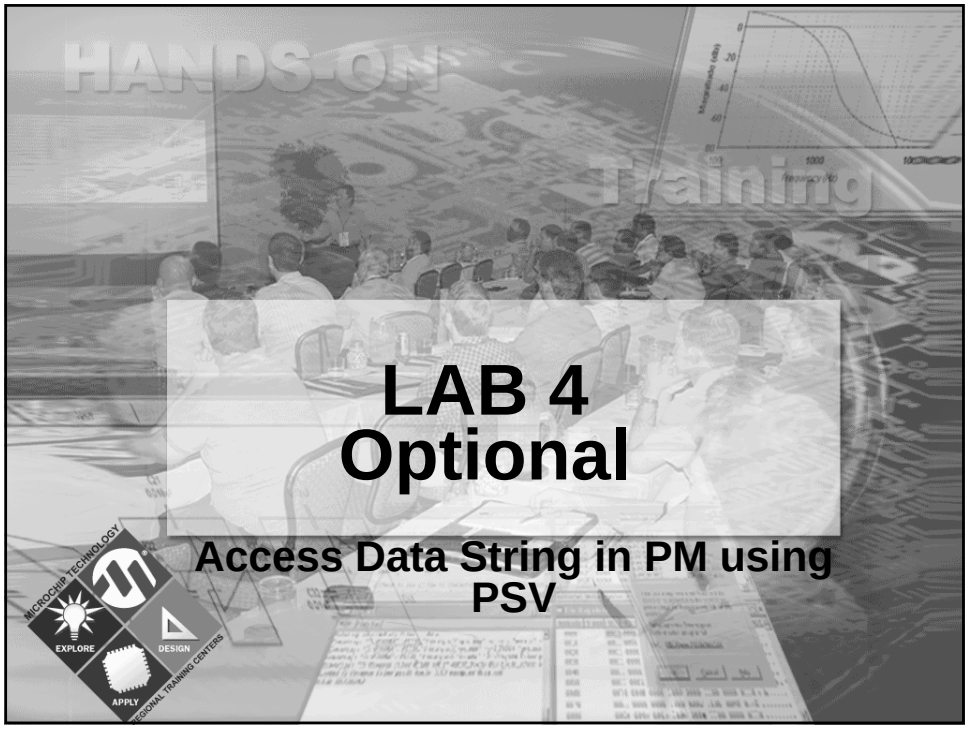
- **PSV allows very large tables of data to be stored and accessed quickly & efficiently**
- **PSV provides a bridge to a common data/program address space (Von Neumann) but only when needed**
- **Example:**
  - Large constant data for display
  - Sine Table

 Class Slide 115

Data Access Overhead Using PSV

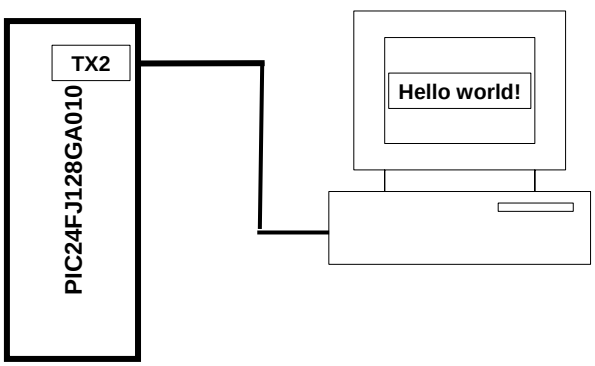
- **PSV data fetch overhead:**
  - Outside a REPEAT loop:
    - **Data move ops, overhead = 1 cycle**
    - **ALU based ops, overhead = 2 cycles**
  - Within a REPEAT loop:
    - **Data pipelined, so overhead for all ops = 0 cycles**
    - **First & last iteration - data pipeline fill/flush**
      - Data move ops, overhead = 1 cycle
      - ALU based ops, overhead = 2 cycles

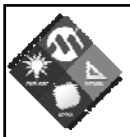
 Class Slide 116



Lab 4: The Task

- Use PSV to access Data String
- Transmit to Hyper Terminal when S3 pressed
- Data String: “Hello world!”





Hints for Lab 4

- Defining constants in PM for PSV, in assembly

```
    .section .const, psv
hello:
    .ascii "Hello World!\n"
```

- Accessing PSV in assembly

```
mov       #psvpage(hello),w0
mov       wo, PSVPAG    ; set PSVPAG address
bset.b    CORCONL,#PSV  ; enable PSV operation
mov       #psvoffset(hello),w0
mov.b     [w0++],w1     ;get first byte

mov       w1,U2TXBUF
```


Summary

- Learned 16-bit Architecture Basics
- Learned the Instruction Set
- Wrote three programs using the Assembly Language
- Brief overview of the Interrupts and Timer1 peripheral
- Leaned about Program Memory Access
- Ready to take the advanced 16-bit Programming Classes

References

- www.microchip.com/16bit
- dsPIC30F Programmer's Reference Manual – DS70030F
- dsPIC30F Family Reference Manual – DS70064C