

## Chapter 2 PIC

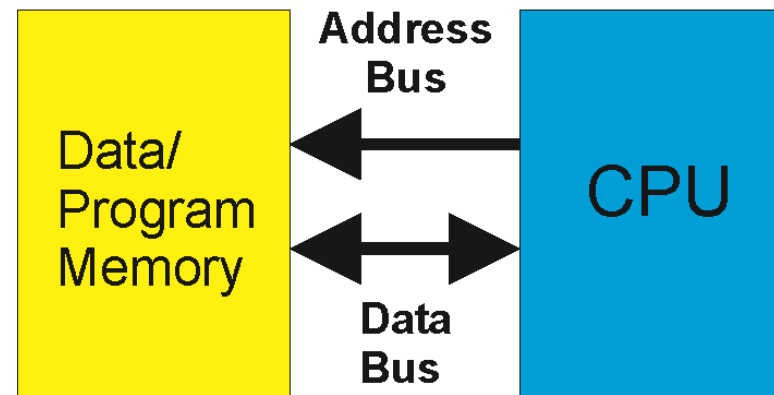
- \* computer architecture
- \* microprocessor, microcontroller, embedded system
- \* PIC18
- \* assembly language
- \* Integrated Development Environment (IDE)

## 2.1 Computer architecture

- computer architecture refers to structure, function, and implementation of computer system
- most computers have operations encoded in binary, stored in memory and performed in a defined sequence – stored program computer

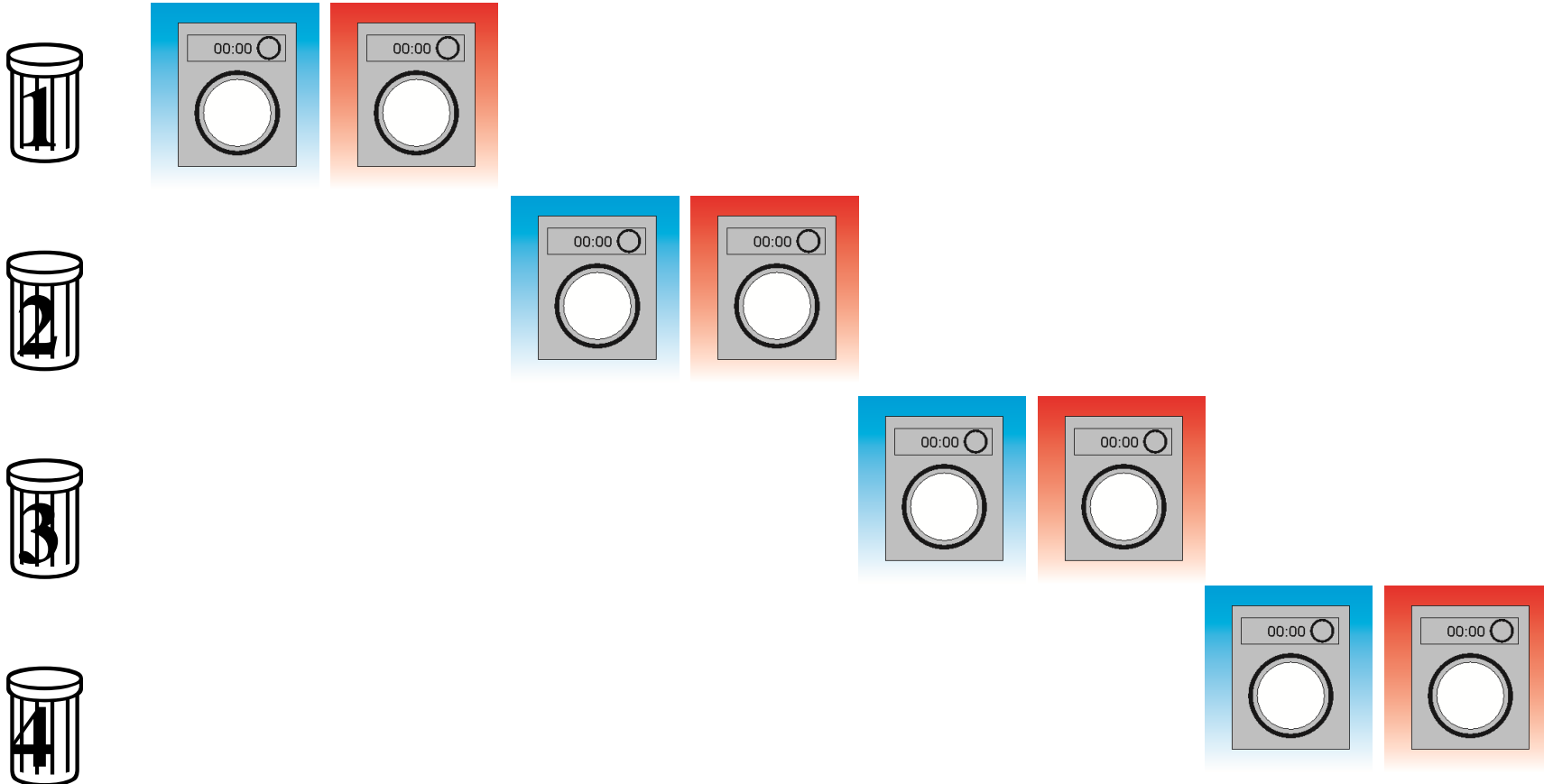
## 2.1.1 von Neumann architecture

- same (logical) memory used for holding both instructions and data
- uses the same address and data bus for ROM space and RAM space
- slow because ROM and RAM cannot be accessed simultaneously



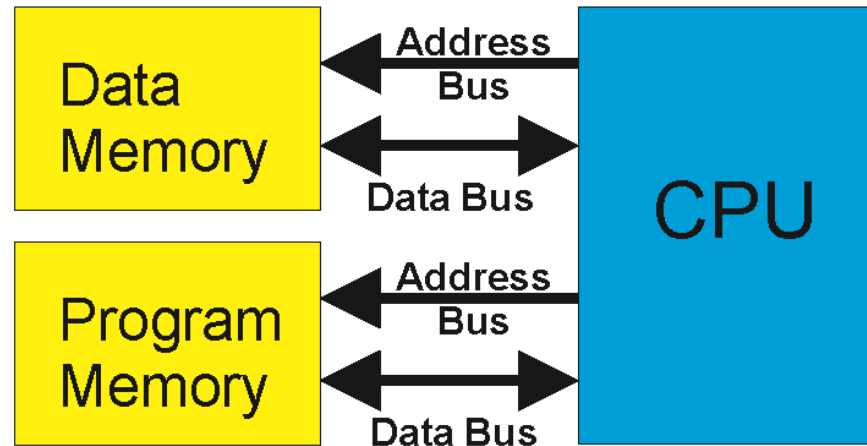
# von Neumann architecture is slow

30 30 30 30 30 30 30 30



## 2.1.2 Harvard architecture

- separated program memory and data memory
- uses different address and data buses to access ROM space and RAM space
- implementation is expensive



# Advantage of Harvard Architecture

(easy for pipeline implementation)

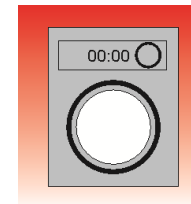
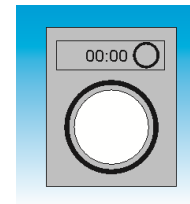
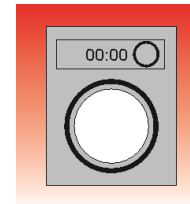
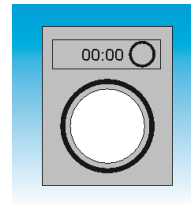
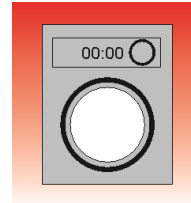
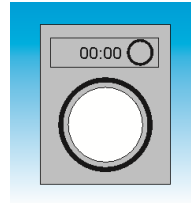
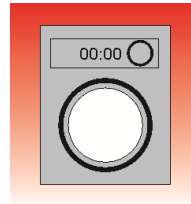
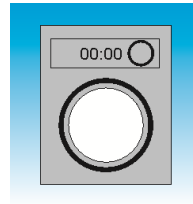
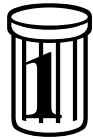
**30**

**30**

**30**

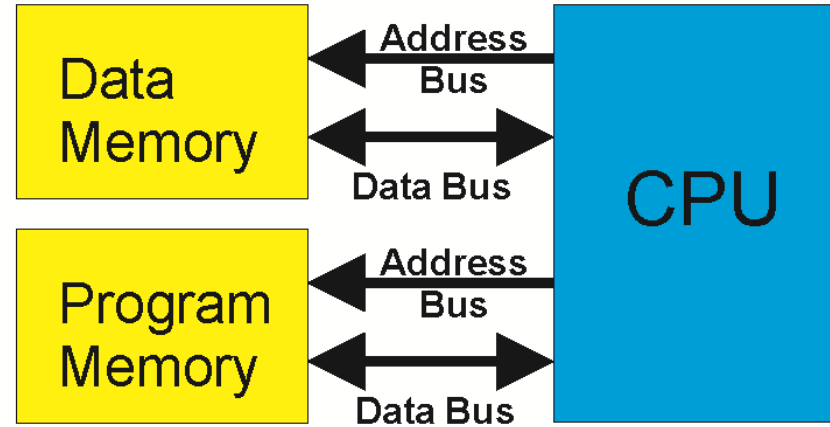
**30**

**30**



- programs are normally only read and not altered during execution
- data might be read or altered
- instructions and data can be brought into the processor simultaneously with separate memories

# Disadvantage of Harvard Architecture



- 64-bit CPU with 64-bit address bus and 32-bit data bus needs about 200 wires
- not efficient use of memory

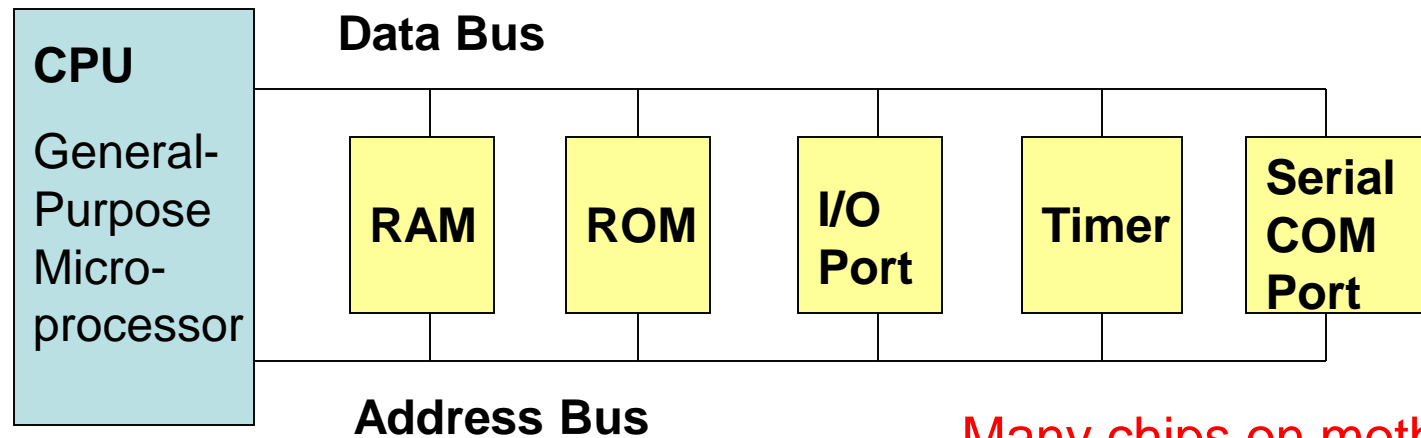


## 2.2 Microprocessor, microcontroller, embedded system

- you certainly know some microprocessors (also called general-purpose microprocessors)
- Can you name one of them?
- in this course, we use microcontroller
- take a look at their similarities and differences

## 2.2.1 general-purpose microprocessor

- CPU for computers
- no RAM, ROM, I/O ports on the CPU chip itself

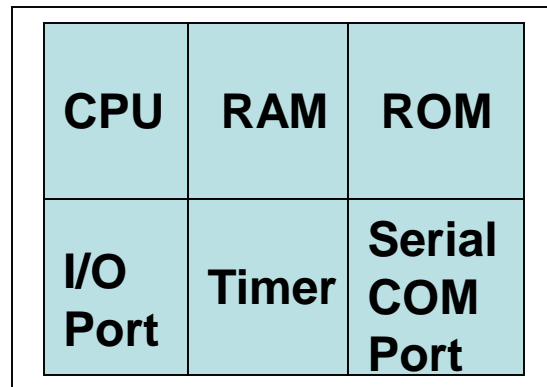


Many chips on mother board

- enable the designer to decide on the amount of ROM, RAM, and I/O ports needed
- adding external ROM, RAM, and I/O ports makes the computer system bulkier and more expensive

## 2.2.2 microcontroller (MCU)

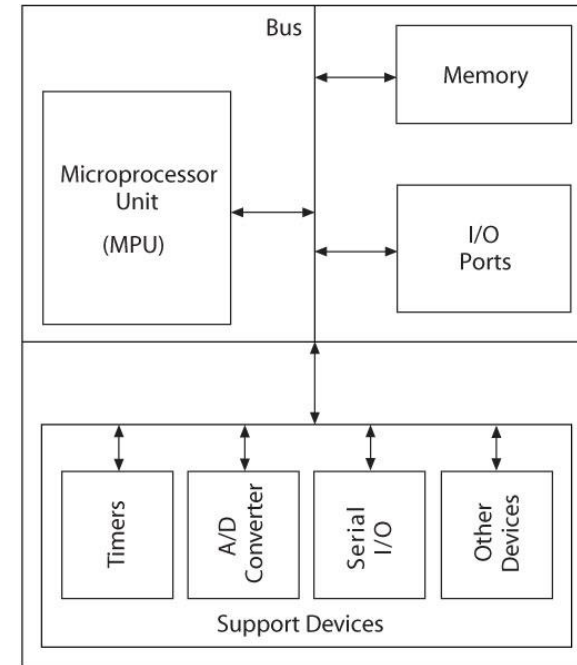
- has a CPU
- on-chip RAM, ROM, I/O ports...
- examples: Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC18



A single chip

MCU is an integrated electronic computing device that includes three major components and some peripheral components on a single chip

- microprocessor
- memory
- I/O ports
- analog-to-digital converter
- digital-to-analog converter
- timers
- pulse width modulation (PWM)
- serial communication interface
- more ....



all components are connected by common communication lines called system bus

MCU have been used in many applications that require computation power:

printer

display

refrigerator

washing machine

microwave oven

car

...

# Microprocessor v.s. Microcontroller

## Microprocessor

- CPU is stand-alone, RAM, ROM, I/O, timer are separate
- Designer can decide on the amount of ROM, RAM and I/O ports
- General-purpose
- Expensive

## Microcontroller

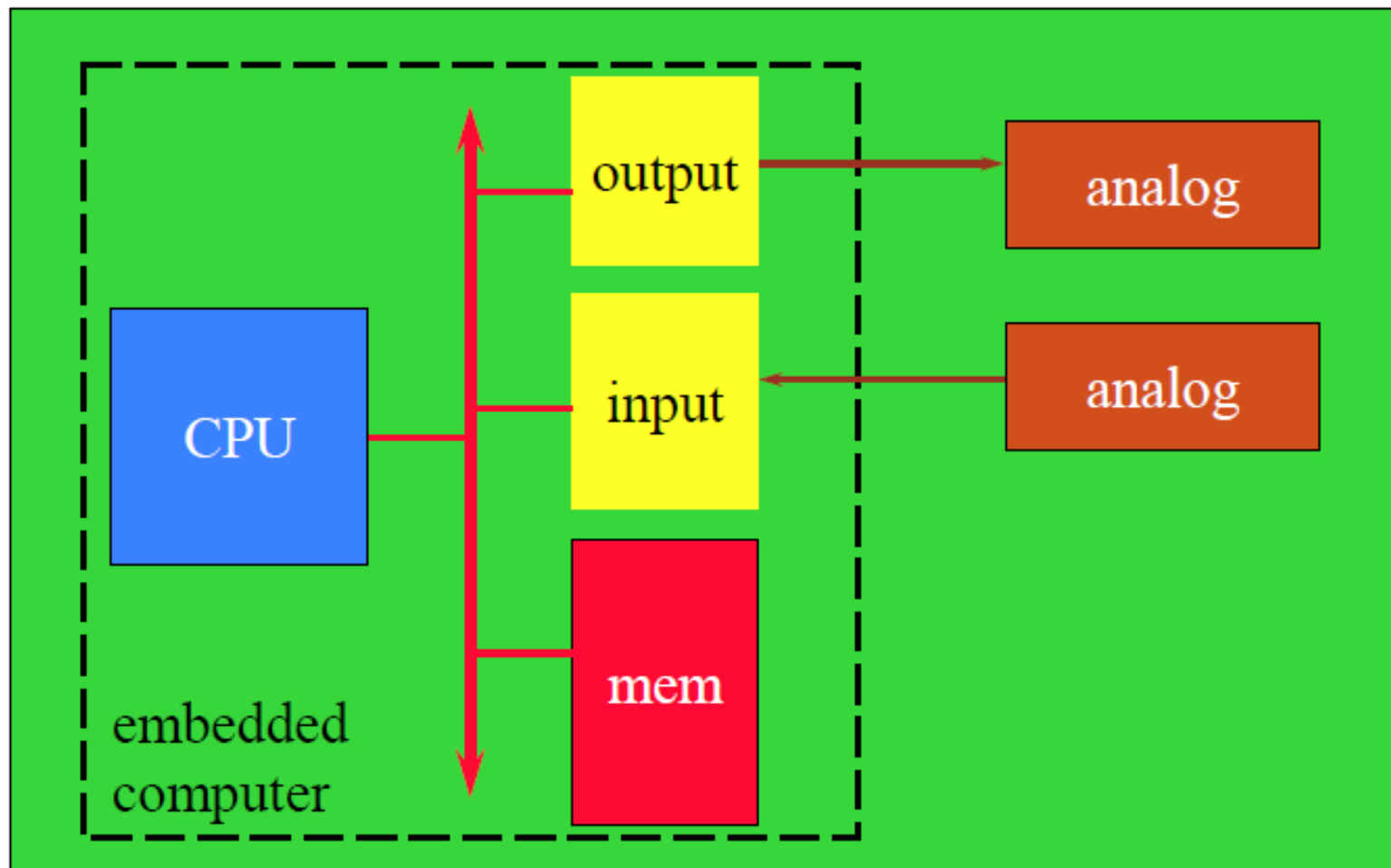
- CPU, RAM, ROM, I/O and timer are all on a single chip
- Fix amount of on-chip ROM, RAM, I/O ports
- Single-purpose
- Cheap
- For applications in which cost, power and space are critical

## 2.2.3 embedded system

- Embedded system means the processor is **embedded into** that application.
- An embedded product uses a microprocessor or microcontroller to **do specified tasks** only.
- In an embedded system, there is only one application software that is typically **burned into ROM**.
- Some embedded products using microcontrollers : printer, keyboard, video game player, door/light control, security system, camera, smartphone, microwave oven, toys ...







# Processors in Embedded Systems

- Which is your choice for an embedded product?
- microcontroller
  - lower cost, lower power consumption, smaller size
  - embedded processor = microcontroller
- microprocessor
  - some applications demand high-performance computing power
  - high-end embedded systems may use microprocessor
  - shorten software development time, save cost
  - in future, an entire computer on a chip

# Three criteria in choosing a microcontroller

1. meeting the computing needs of the task efficiently and cost effectively
  - speed of MCU
  - package size
  - power consumption (use battery)
  - the amount of ROM and RAM
  - the number of I/O ports and timers
  - easy to upgrade (need to re-write the software?)
  - cost per unit

## 2. availability of software development tools

- assembler
- debugger
- C language compiler
- emulator
- simulator
- technical support

## 3. wide availability and reliable sources of the microcontroller

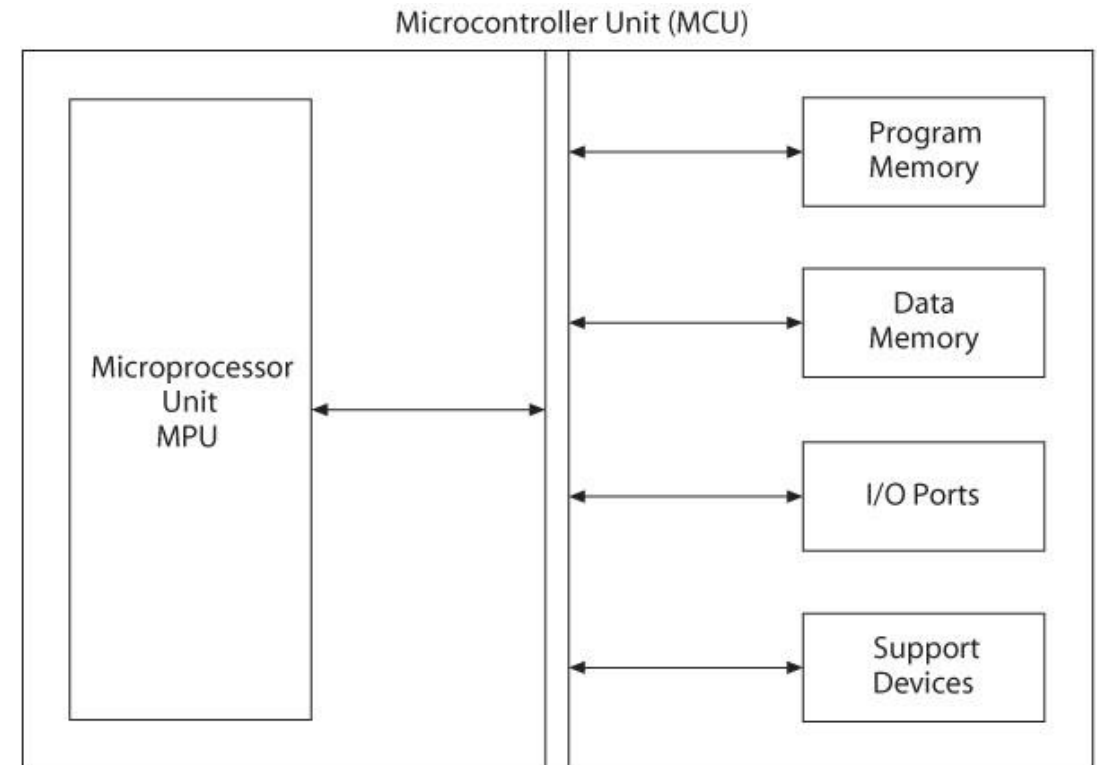
## 2.3 PIC18

- in 1989, Microchip Technology Corporation introduced an 8-bit microcontroller called Peripheral Interface Controller (PIC)
- the MCU had a small ROM for program, a small RAM for data, 1 timer, few I/O ports, all on a single chip

- Microchip has introduced 6 lines of 8-bit MCUs
  - PIC12XXX: 12- or 14-bit instruction
  - PIC14000: 14-bit instruction
  - PIC16C5X: 12-bit instruction
  - PIC16CXX: 14-bit instruction
  - PIC17: 16-bit instruction
  - PIC18: 16-bit instruction

Software is not 100% compatible!

- PIC microcontrollers are designed using the Harvard architecture which includes:
  - Microprocessor unit (MPU)
  - Program memory for instructions
  - Data memory for data
  - I/O ports
  - Support devices such as timers



## 2.3.1 microprocessor unit

- Includes Arithmetic Logic Unit (ALU), Registers, and Control Unit
  - Arithmetic Logic Unit (ALU)
    - Instruction decoder
      - 16-bit instructions
    - Status register that stores flags
      - 5-bits
    - working register (WREG)
      - 8-bit accumulator



- Registers
  - Program Counter (PC)
    - 21-bit register that holds the program memory address while executing programs
  - Bank Select Register (BSR)
    - 4-bit register used in **direct addressing** the data memory
  - File Select Registers (FSRs)
    - 12-bit registers used as memory pointers in **indirect addressing** the data memory
- Control unit
  - provides timing and control signals for Read and Write operations

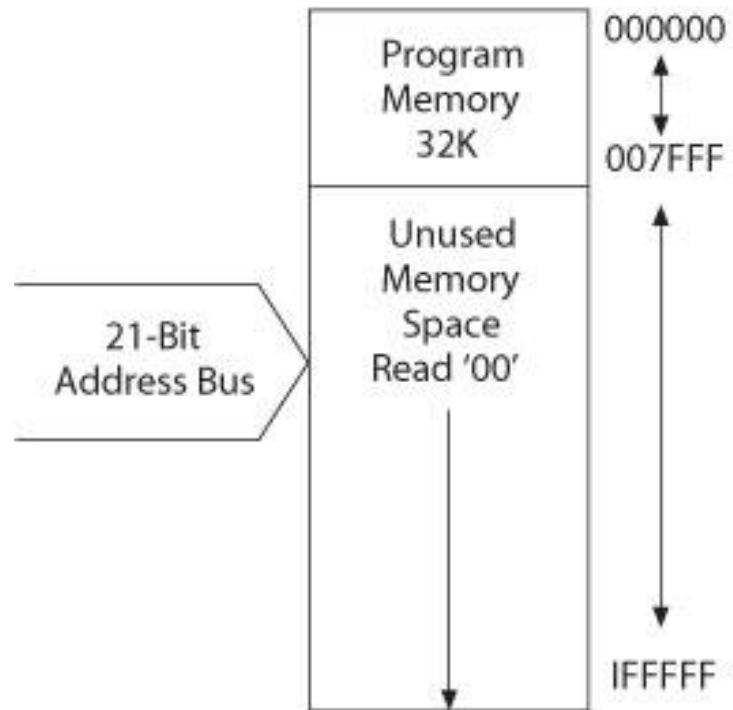
## 2.3.2 system bus

- Address bus
  - 21-bit address bus for program memory, addressing capacity 2 MB
  - 12-bit address bus for data memory, addressing capacity 4 KB
- Data bus
  - 16-bit instruction/data bus for program memory
  - 8-bit data bus for data memory
- Control signals
  - Read and Write

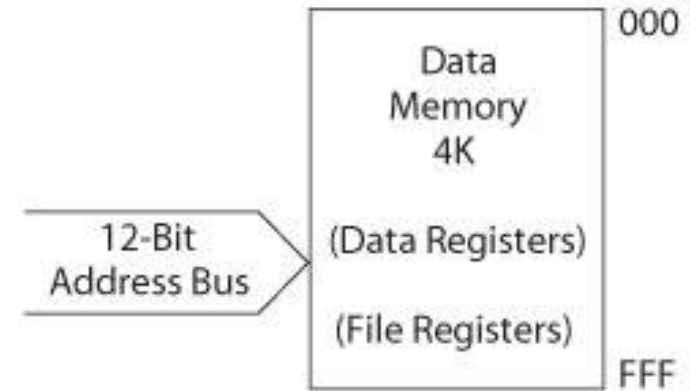
### 2.3.3 memory

- Program Memory: 32 KB
  - Address range: 000000 to 007FFF<sub>H</sub>
- Data Memory: 4 KB
  - Address range: 000 to FFF<sub>H</sub>
- Data EEPROM
  - Not part of the data memory space
  - Addressed through special function registers
  - Used mainly for storage of critical data (does not need to be changed very often)

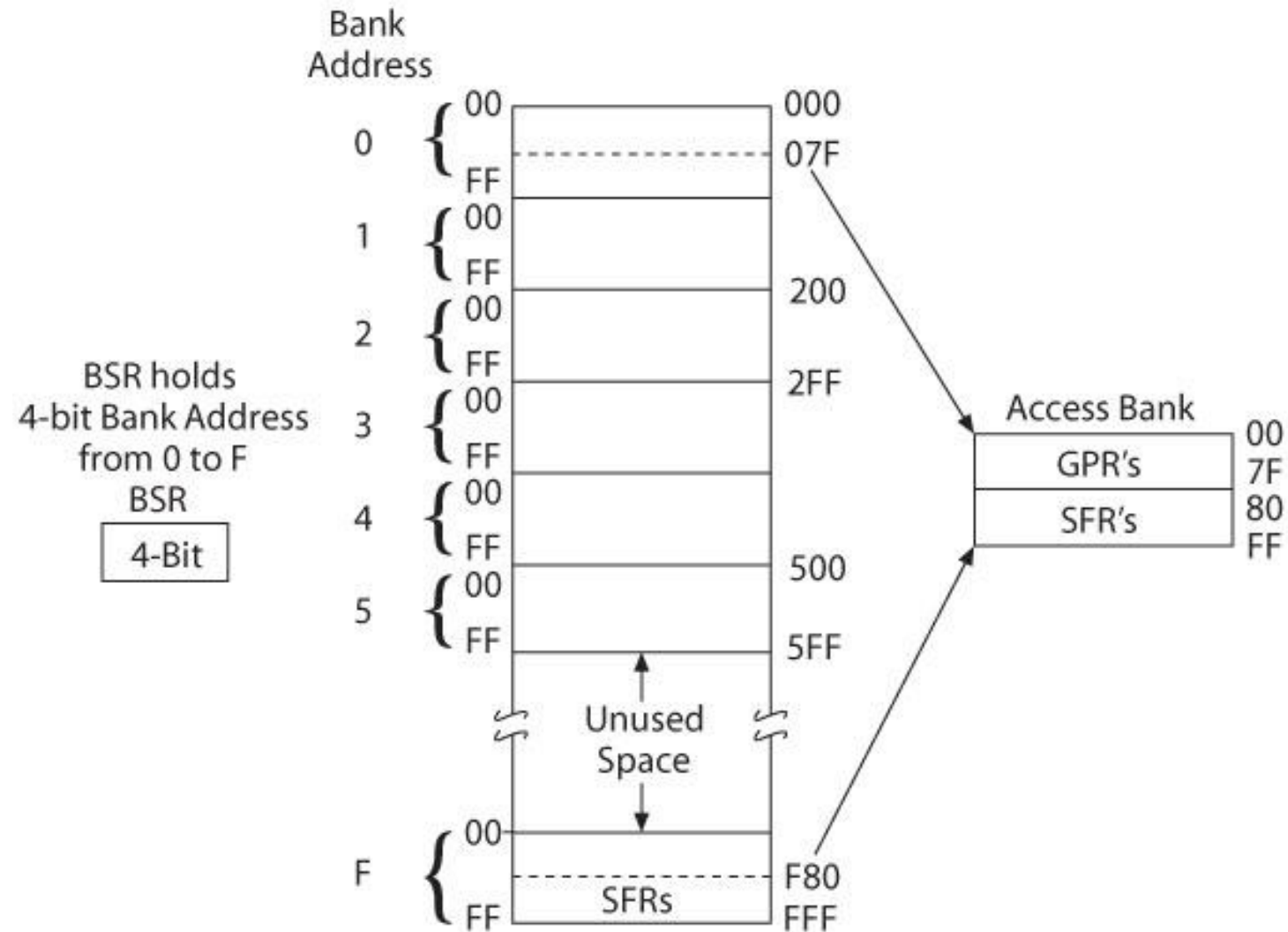
## Program memory



## Data memory



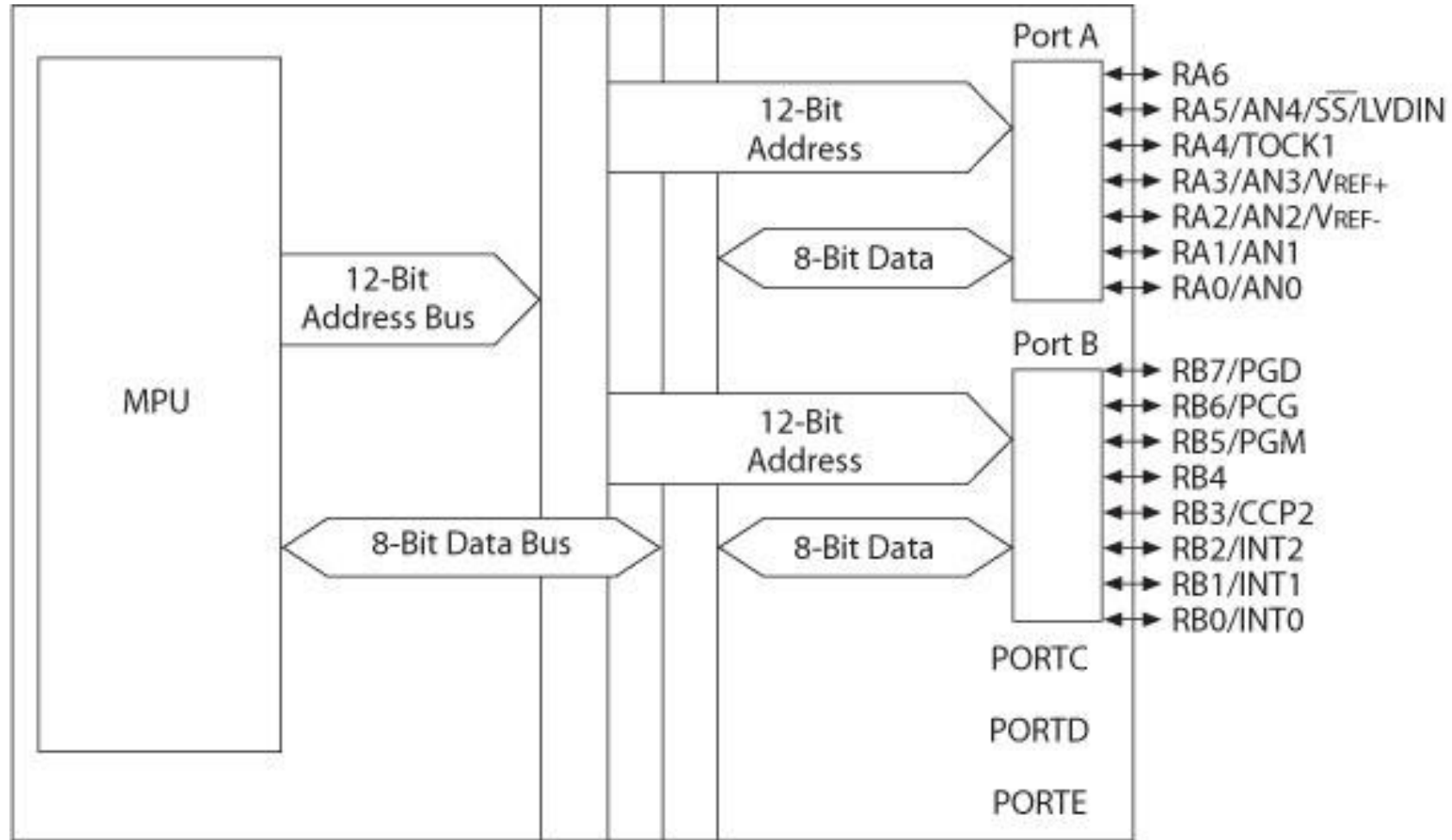
# PIC18F452/4520 – Data Memory Map



## 2.3.4 I/O and support devices

- Five I/O ports
  - Digital IO is the most fundamental mode of connecting a MCU to external world. The interface is done using what is called a PORT.
  - The IO ports can be programmed as input or output.
  - PORT A through PORT E
  - Most I/O pins are multiplexed
  - Generally have eight I/O pins with a few exceptions
  - Addresses already assigned to these ports in the design stage
  - Each port is identified by its assigned SFR

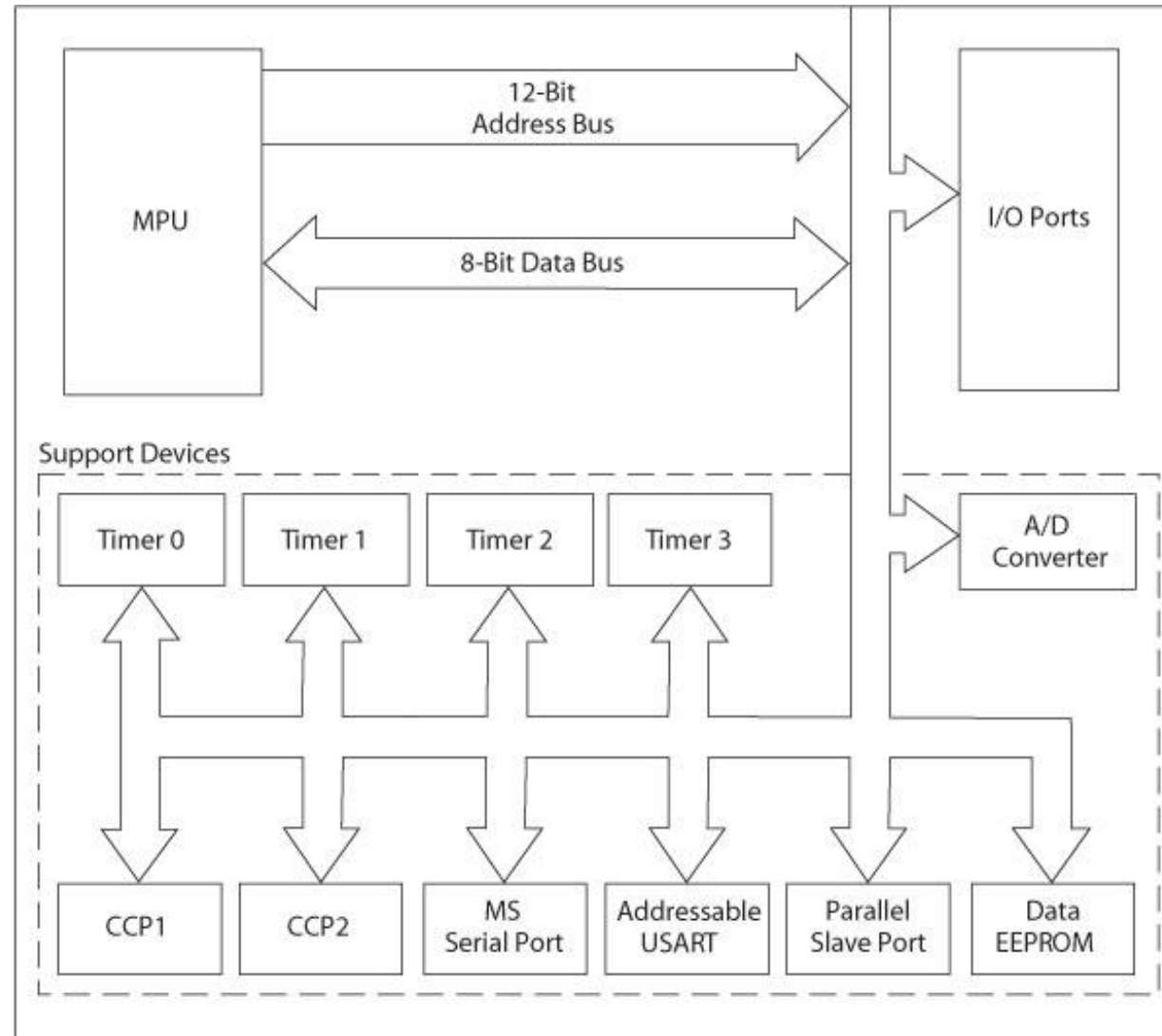
# PIC18F452—I/O Ports A and B



# MCU Support Devices

- Timers
  - counter, interrupt
- CCP
  - Capture, Compare, **P**ulse width modulation
- Serial Communications
  - Master Synchronous Serial Port (MSSP)
  - Addressable USART
- Parallel Slave Port (PSP)
- Analog-to-digital converter (ADC)
- Data EEPROM



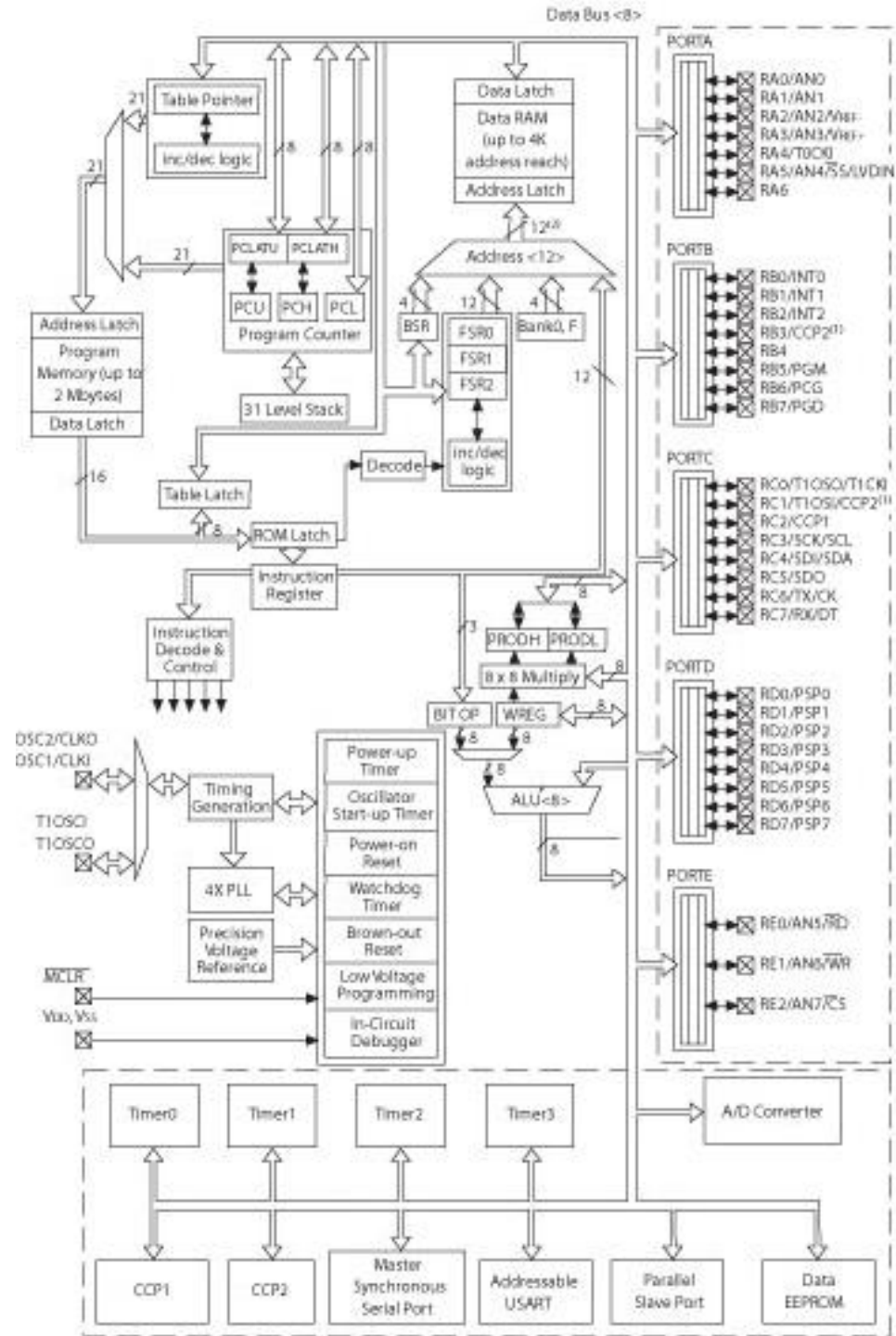


### 2.3.5 special features

- Sleep (low-power operation) mode
- Watchdog timer (WDT)
- Code protection
- In-circuit serial programming
- In-circuit debugger

# PIC18F4X2

## Architecture Block Diagram



## List of Selected Microcontroller Families from Microchip

Part No.	Program OTP/Flash	EE PROM	RAM	Total Pins	I/O Pins	ADC	Analog Comp.	Digital Timers/ WDT	Serial I/O	CCP/ ECCP	Max Speed MHz	Instruc- tion Size	Total Instruc- tions
10F200	256x12 Flash		16	8	4			1-8 bit, 1-WDT			4	12-bit	33
10F220	256x12 Flash		16	8	4	2x8-bit		1-8 bit, 1-WDT			8	12-bit	33
12F510	1536x12 Flash		38	8	6	3x8-bit	1	1-8 bit, 1-WDT			8	12-bit	33
16F506	1536x12 Flash		67	14	12	3x8-bit	2	1-8 bit, 1-WDT			20	12-bit	33
16C55A	768x12 OTP		24	28	20			1-8 bit, 1-WDT			40	12-bit	33
16CR58B	3072x12 ROM		73	18	12			1-8 bit, 1-WDT			20	12-bit	33
12F683	2048x14 Flash	256	128	8	6	4x10-bit	1	1-16 bit, 2-8 bit, 1-WDT			20	14-bit	35
16F687	2048x14 Flash	256	128	20	18	12x10- bit	2	1-16 bit, 1-8 bit, 1-WDT	EU/I <sup>2</sup> C/ SPI		20	14-bit	35
18F1230	2048x16 Enh Flash	128	256	18-28	16	4x10-bit	3	2-16 bit, 1-WDT	EU		40	16-bit	77
18F4520	16384x16 Enh Flash	256	1536	40-44	36	13x10- bit	2	1-8 bit, 3-16 bit, 1-WDT	EU/ MI <sup>2</sup> C /SPI	1/1	40	16-bit	77
18F6527	24576x16 Enh Flash	1024	3936	64	54	12x10- bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI <sup>2</sup> C /SPI	2/3	40	16-bit	77
18F8622	32768x16 Enh Flash	1024	3936	80	70	16x10- bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI <sup>2</sup> C /SPI	2/3	40	16-bit	77
18F96J60	32768x16 Flash		2048	100	72	16x10- bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI <sup>2</sup> C /SPI	2/3	42	16-bit	77
24FJ128GA-010	65536x16 Flash		8192	100- 128	86	16x10- bit	2	5-16 bit, 1-WDT	2 -UART 2-I <sup>2</sup> C/ SPI	5	32	16-bit	77

Abbreviations: 1) ADC: Analog-Digital Converter, 2) AUSART: Addressable USART, 3) CCP: Capture/Compare/PWM, 4) ECCP: Enhanced CCP,

5) EU: Enhanced USART, 6) Enh Flash: Enhanced Flash, 7) I<sup>2</sup>C: Inter-integrated Circuit Bus, 8) MI<sup>2</sup>C/SPI: Master I<sup>2</sup>C /SPI, 9) OTP: One-Time Programmable,

10) SPI: Serial Peripheral Interface, 11) USART: Universal Synchronous/Asynchronous Receiver/Transmitter, 11) WDT: Watchdog Timer

## 2.4 Assembly language

- 77 instructions
- In PIC18F instruction set, all instructions are 16-bit word length except four instructions that are 32-bit length

# Instructions

- Copy (Move) 8-bit number (Literal) into WREG
  - Mnemonics: MOVLW 8-bit
  - Binary format:  
0000 1110 XXXX XXXX (any 8-bit number)
- Copy (Move) contents of WREG into PORTC (File)
  - Mnemonics: MOVWF PORTC, a
    - ('a' indicates that PORTC is in the Access Bank)
  - Binary format:  
0110 1110 1000 0010 (82<sub>H</sub> is PORTC address)

# **Illustrative Program: Displaying a byte at an I/O port**

- Problem statement:
  - Write instructions to light up alternate LEDs at PORTC
- Hardware:
  - PORTC
    - bidirectional (input or output) port; should be setup as output port for display
  - Logic 1 will turn on an LED

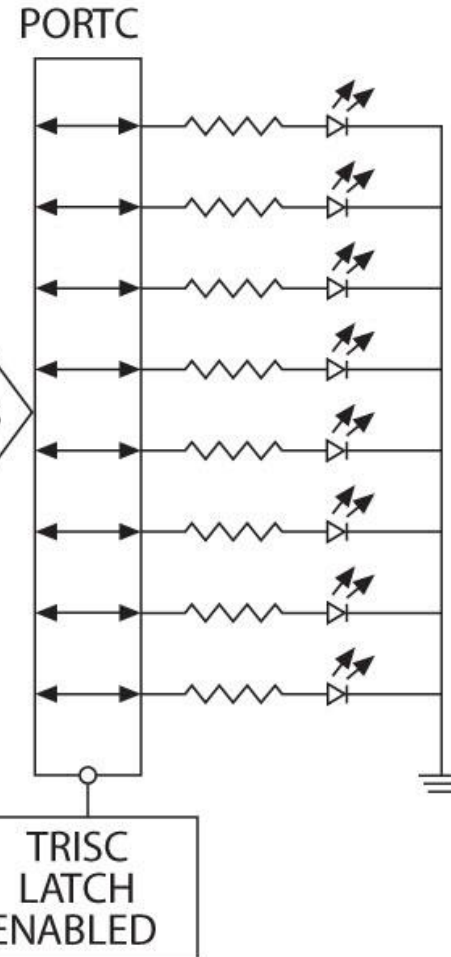
- Interfacing LEDs to PORTC

**0x55**

**0  
1  
0  
1  
0  
1  
0  
1**

from  
MPU

8-Bit Data Bus

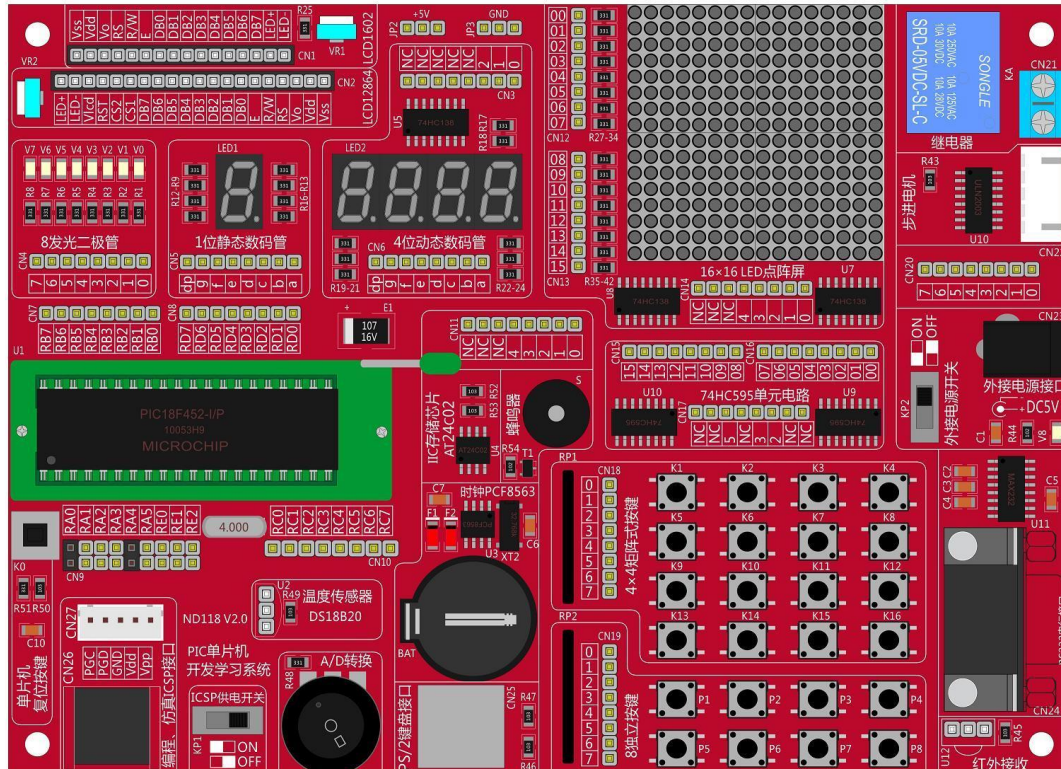




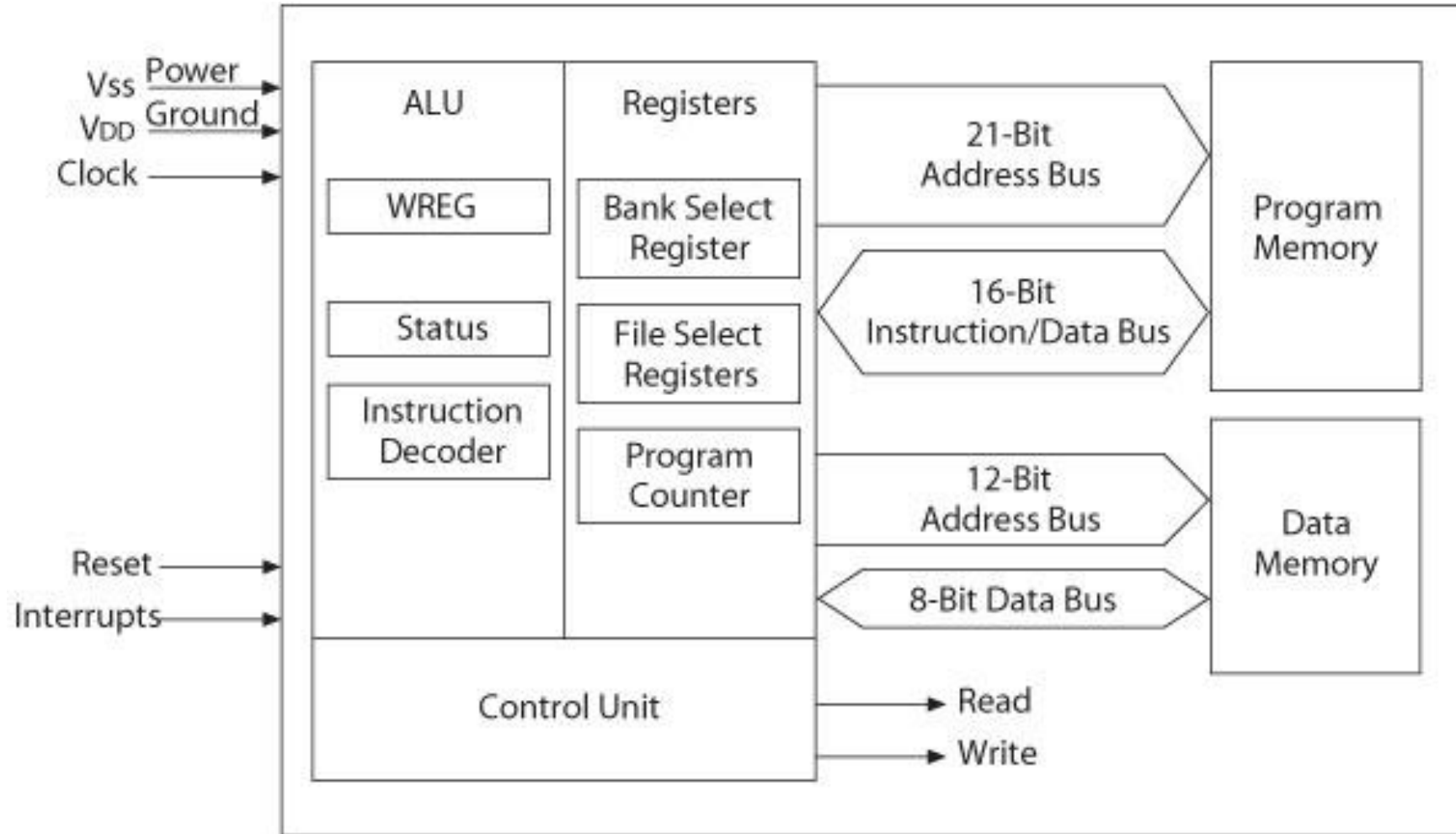
- Program (software)
  - Logic 0 to TRISC sets up PORTC as an output port
  - Byte 55<sub>H</sub> turns on alternate LEDs
    - MOVLW        00                    ;Load W register with 0
    - MOVWF        TRISC,0            ;Set up PORTC as output
    - MOVLW        0x55                ;Byte 55<sub>H</sub> to turn on LEDS
    - MOVWF        PORTC,0            ;Turn on LEDs

## 2.5 IDE

ND118-452 Development Kit is a single board multi-function PIC18F4520 microcontroller platform.



# PIC18F4520



# MPLAB IDE V8

- For you to develop the program for PIC18.
- Change the assembly program to machine code.
- Allow you to simulate the program in PC.
- Allow you download your machine code to ND118-452 Development Kit. (We need a device called PICkit3)
- Allow you control the program running in ND118-452 Development Kit.

You can download the IDE from

<http://www.microchip.com/development-tools/downloads-archive>

Note : you should select **MPLAB IDE v8.XX** versions rather than **MPLAB IDE X**.

# Program Example

Action	Machine Code
Move value 21H into register W	0E21
Add value 42H to register W	0F42
Add value 12H to register W	0F12
No operation	0000
GOTO 0000	EF00 F000

```
Main:  ORG          0x0000
        MOVLW     0x21
        ADDLW     0x42
        ADDLW     0x12
        NOP
        GOTO      Main
        END
```

expt1 - MPLAB IDE v8.84

File Edit View Project Debugger Programmer Tools Configure Window Help

Debug Checksum: 0x77dd

Program Memory

Line	Address	Opcode	
1	0000	0E21	MOVLW 0x21
2	0002	0F42	ADDLW 0x42
3	0004	0F12	ADDLW 0x12
4	0006	0000	NOP
5	0008	EF00	GOTO 0
6	000A	F000	NOP
7	000C	0000	NOP
8	000E	FFFF	NOP
9	0010	FFFF	NOP
10	0012	FFFF	NOP

Opcode Hex Machine Symbolic

Watch

Add SFR ADCON0 Add Symbol \_BOREN\_NOSLP\_2L

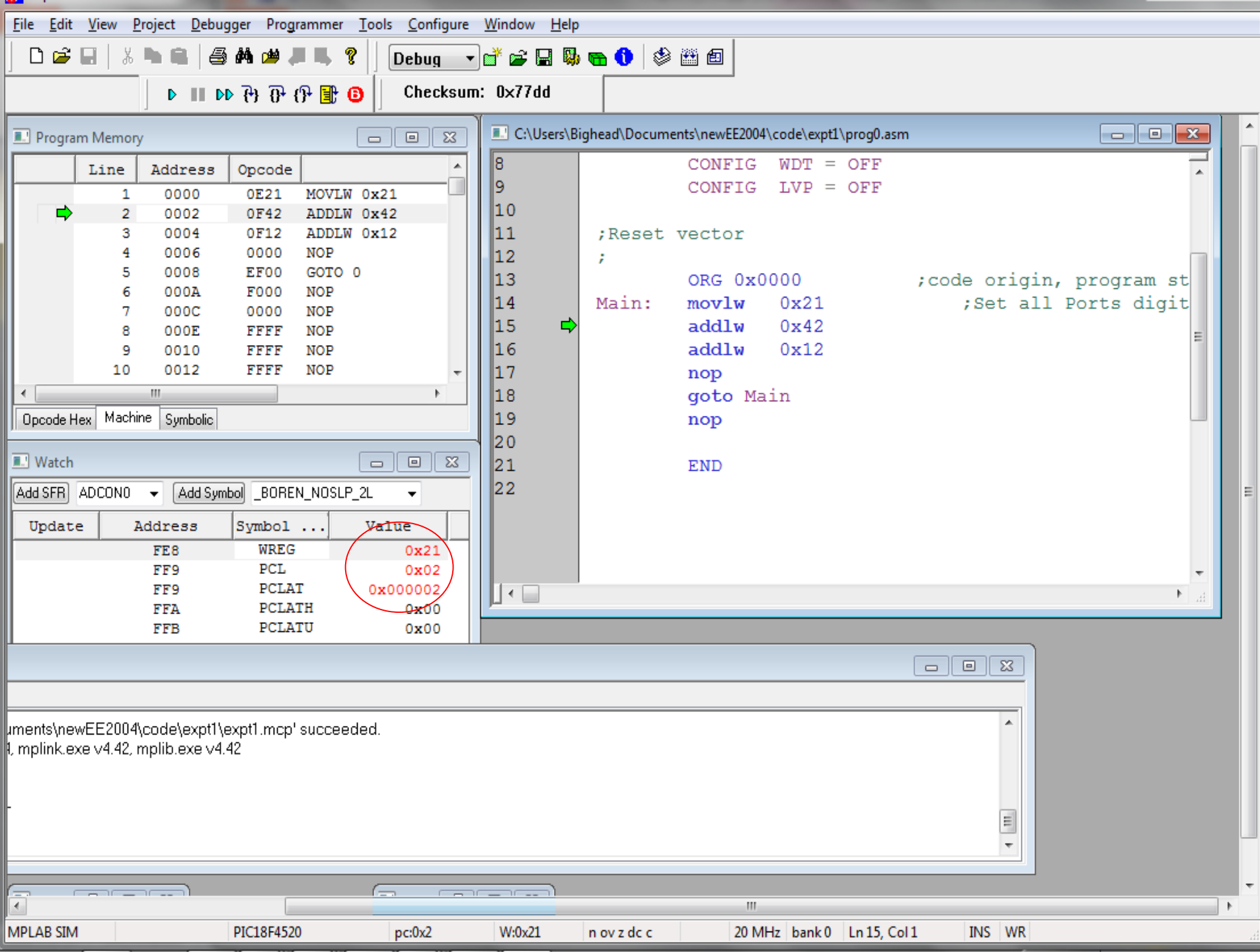
Update	Address	Symbol ...	Value
	FE8	WREG	0x75
	FF9	PCL	0x00
	FF9	PCLAT	0x000000
	FFA	PCLATH	0x00
	FFB	PCLATU	0x00

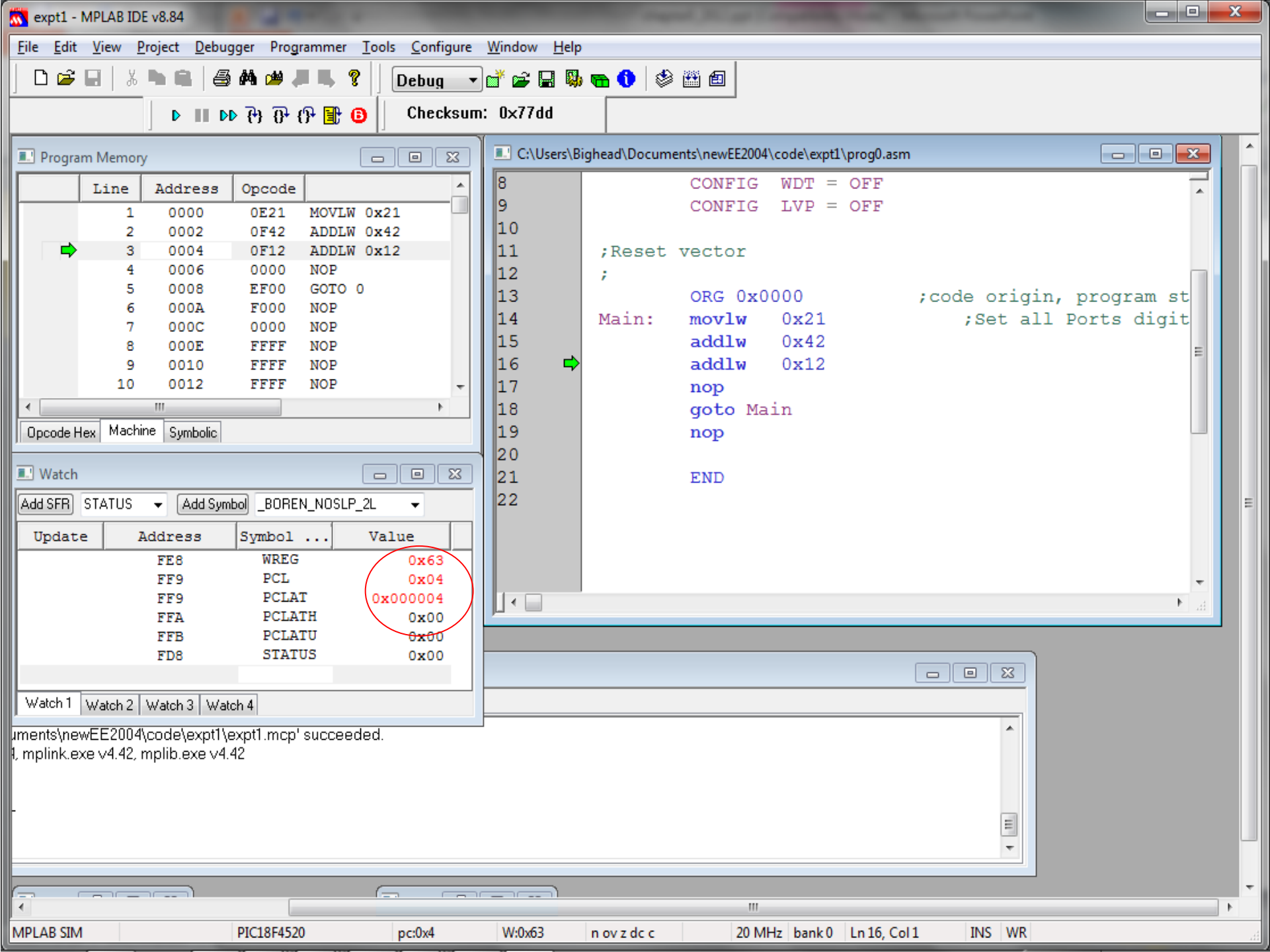
C:\Users\Bighead\Documents\newEE2004\code\expt1\prog0.asm

```
8          CONFIG WDT = OFF
9          CONFIG LVP = OFF
10
11      ;Reset vector
12      ;
13          ORG 0x0000      ;code origin, program start
14      Main: movlw 0x21      ;Set all Ports digit
15          addlw 0x42
16          addlw 0x12
17          nop
18          goto Main
19          nop
20
21          END
22
```

Documents\newEE2004\code\expt1\expt1.mcp' succeeded.  
1, mplink.exe v4.42, mplib.exe v4.42

MPLAB SIM PIC18F4520 pc:0 W:0x75 n ov z dc c 20 MHz bank 0 Ln 14, Col 1 INS WR







expt1 - MPLAB IDE v8.84

File Edit View Project Debugger Programmer Tools Configure Window Help

Debug Checksum: 0x77dd

Program Memory

Line	Address	Opcode	
1	0000	0E21	MOVLW 0x21
2	0002	0F42	ADDLW 0x42
3	0004	0F12	ADDLW 0x12
4	0006	0000	NOP
5	0008	EF00	GOTO 0
6	000A	F000	NOP
7	000C	0000	NOP
8	000E	FFFF	NOP
9	0010	FFFF	NOP
10	0012	FFFF	NOP

Opcode Hex Machine Symbolic

Watch

Add SFR STATUS Add Symbol \_BOREN\_NOSLP\_2L

Update	Address	Symbol ...	Value
	FE8	WREG	0x75
	FF9	PCL	0x06
	FF9	PCLAT	0x000006
	FFA	PCLATH	0x00
	FFB	PCLATU	0x00
	FD8	STATUS	0x00

Watch 1 Watch 2 Watch 3 Watch 4

C:\Users\Bighead\Documents\newEE2004\code\expt1\prog0.asm

```
8          CONFIG WDT = OFF
9          CONFIG LVP = OFF
10
11      ;Reset vector
12      ;
13          ORG 0x0000          ;code origin, program st
14      Main: movlw 0x21          ;Set all Ports digit
15          addlw 0x42
16          addlw 0x12
17          nop
18          goto Main
19          nop
20
21      END
22
```

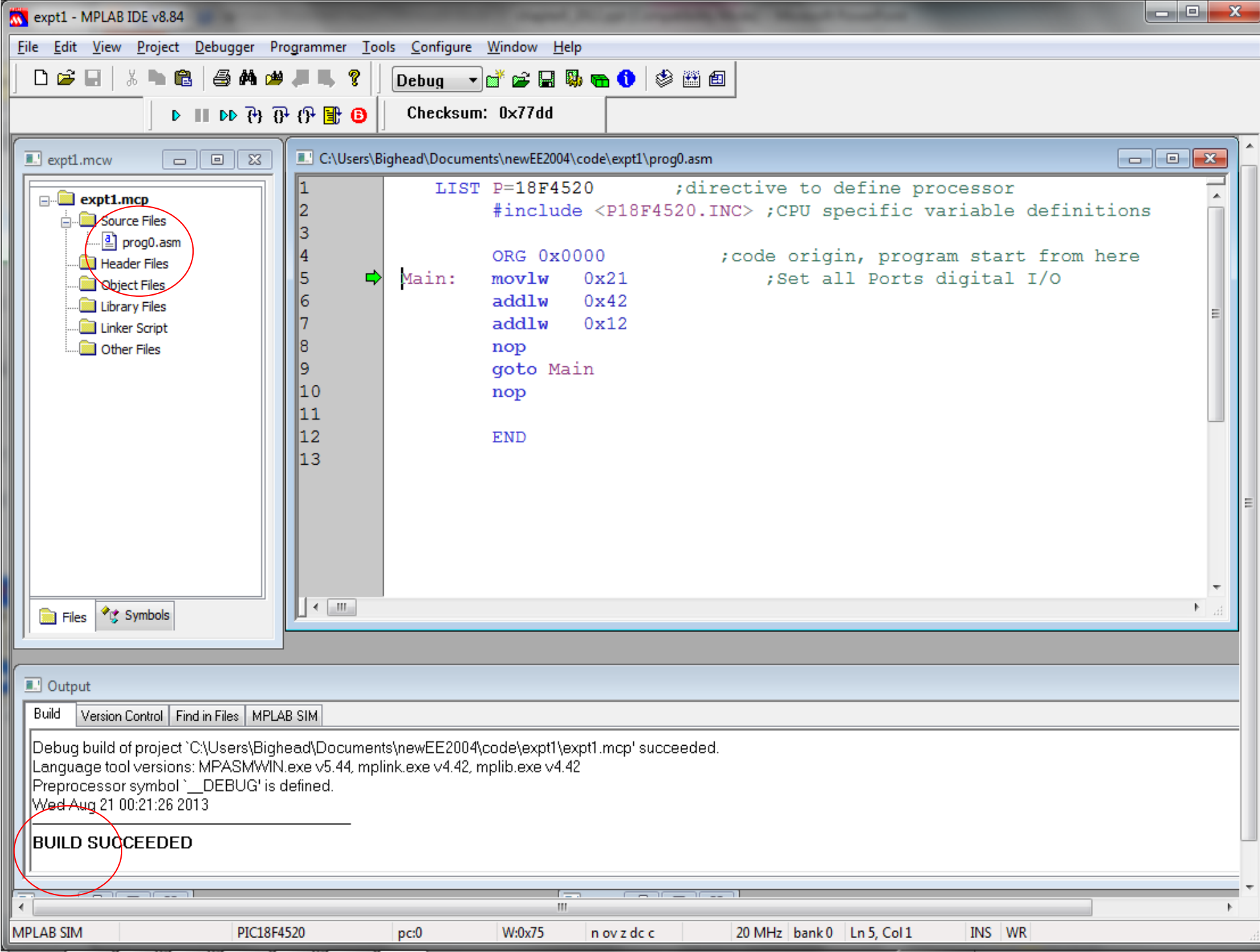
Documents\newEE2004\code\expt1\expt1.mcp' succeeded.  
1. mplink.exe v4.42, mplib.exe v4.42

MPLAB SIM PIC18F4520 pc:0x6 W:0x75 n ov z dc c 20 MHz bank 0 Ln 17, Col 1 INS WR

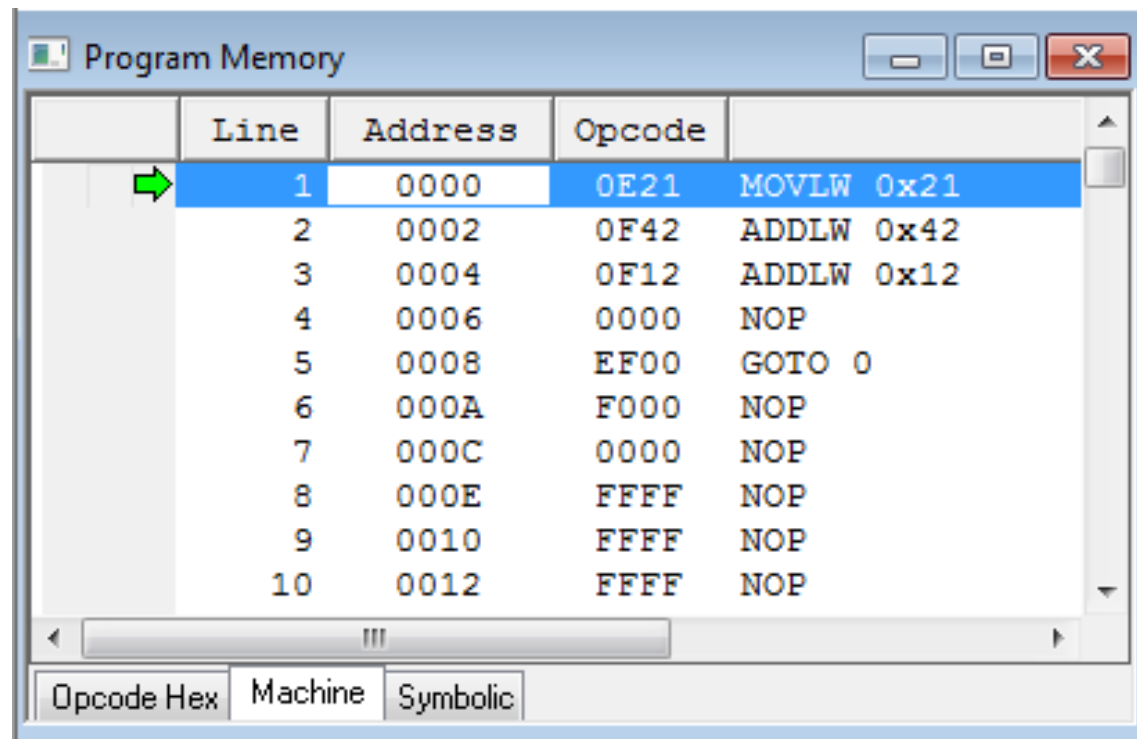
# Procedure

1. Execute the MPLAB IDE program.
2. Click “File”; “New” and type the code into the file editor window.
3. Click “File”, “Save As”. Create and Select the “My Document\Code\Chapter0” folder and type “prog0.asm” as the program file name. (Make sure you save the file into the Chapter0 folder).
4. Click “Project”, “Project Wizard...”, “Next >”, select device “PIC18F4520”, click “Next >”, select “Microchip MPASM Toolsuite”, click “Next >”
5. Browse into folder “My Document\Code\Chapter0”, type “expt1” as the Project file name and click “Save”.
6. Click “Next >”, expand the folder tree and locate the file “prog0.asm”. Click “Add >>” and “Next >” to put the “prog0.asm” file to the Project. Check the project parameters list and click “Finish” to finish the project definition process.

7. Click “Project”, “Build All” and select “Absolute”.
8. “BUILD SUCCEEDED” should appear at Output window.  
Should “BUILD FAILED” appear instead, check for the error messages, fix any errors found and repeat the build process (*step 7*) until success.
9. Click “File”, “Save Workspace” to save your work. It will save all your current project related parameters into the file with extension “mcw”. You can double click file “expt1.mcw” later to continue your development.



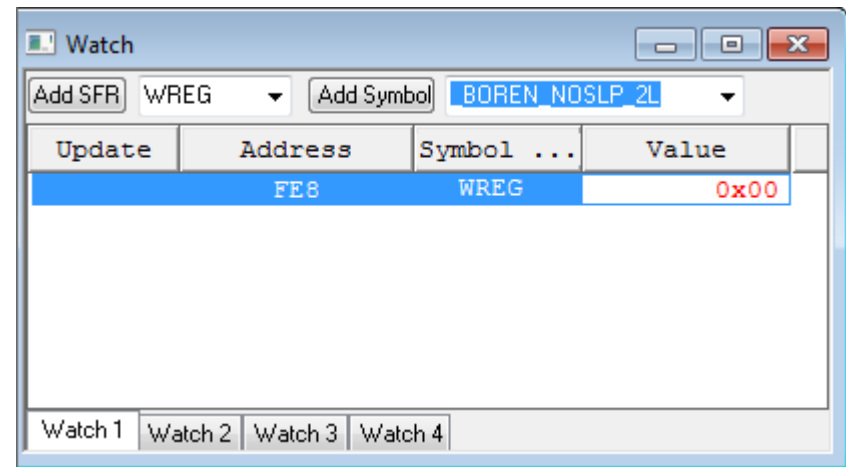
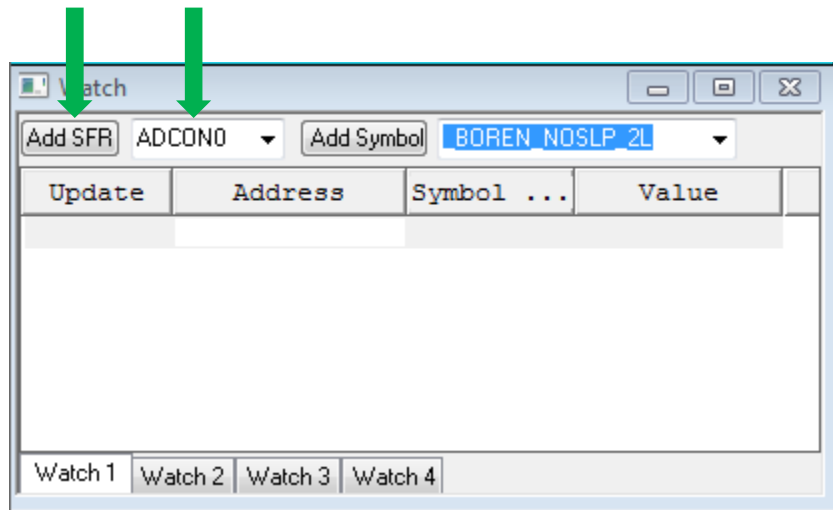
10. Click “Debugger”, “Select Tool”, “4 MPLAB SIM” to select MPSIM simulator. Screen look remain unchanged, however. MPSIM takes the control.
11. Click “View”, “Program Memory”.  
(see the contents of program memory)



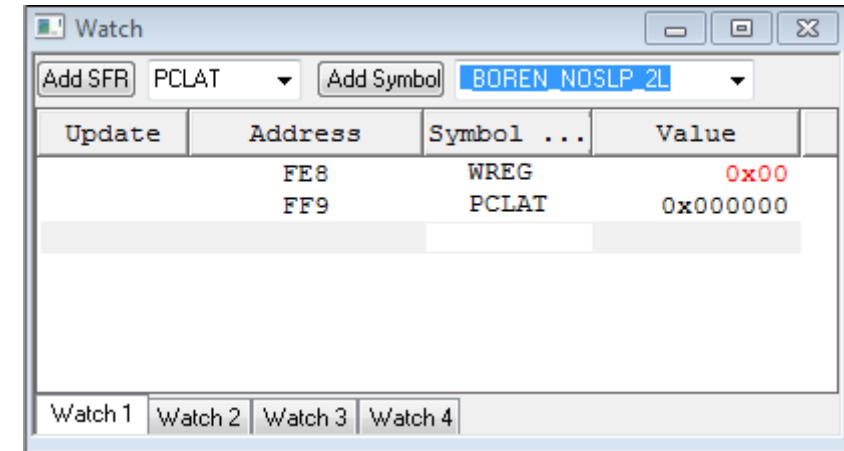
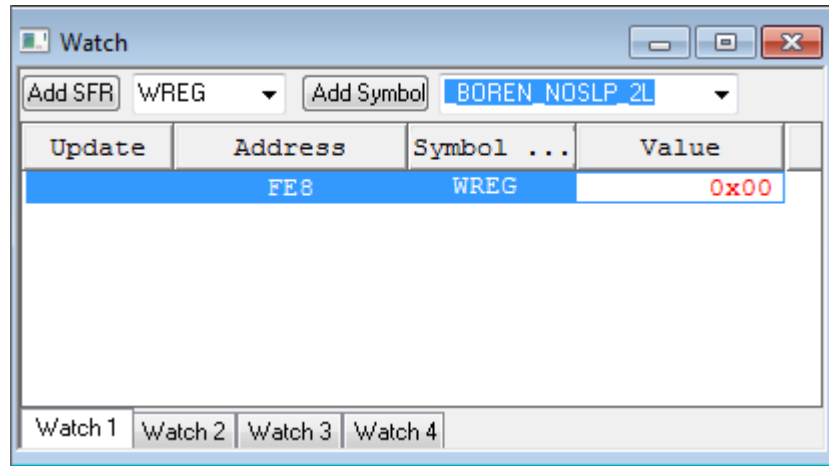
Line	Address	Opcode	Symbolic
1	0000	0E21	MOVLW 0x21
2	0002	0F42	ADDLW 0x42
3	0004	0F12	ADDLW 0x12
4	0006	0000	NOP
5	0008	EF00	GOTO 0
6	000A	F000	NOP
7	000C	0000	NOP
8	000E	FFFF	NOP
9	0010	FFFF	NOP
10	0012	FFFF	NOP

At the bottom of the window, there are three tabs: "Opcode Hex", "Machine", and "Symbolic". The "Symbolic" tab is currently selected.

12. Click “View”, “Watch”. Select “WREG” and then click “Add SFR” button.  
(Now you can watch WREG during the simulation)

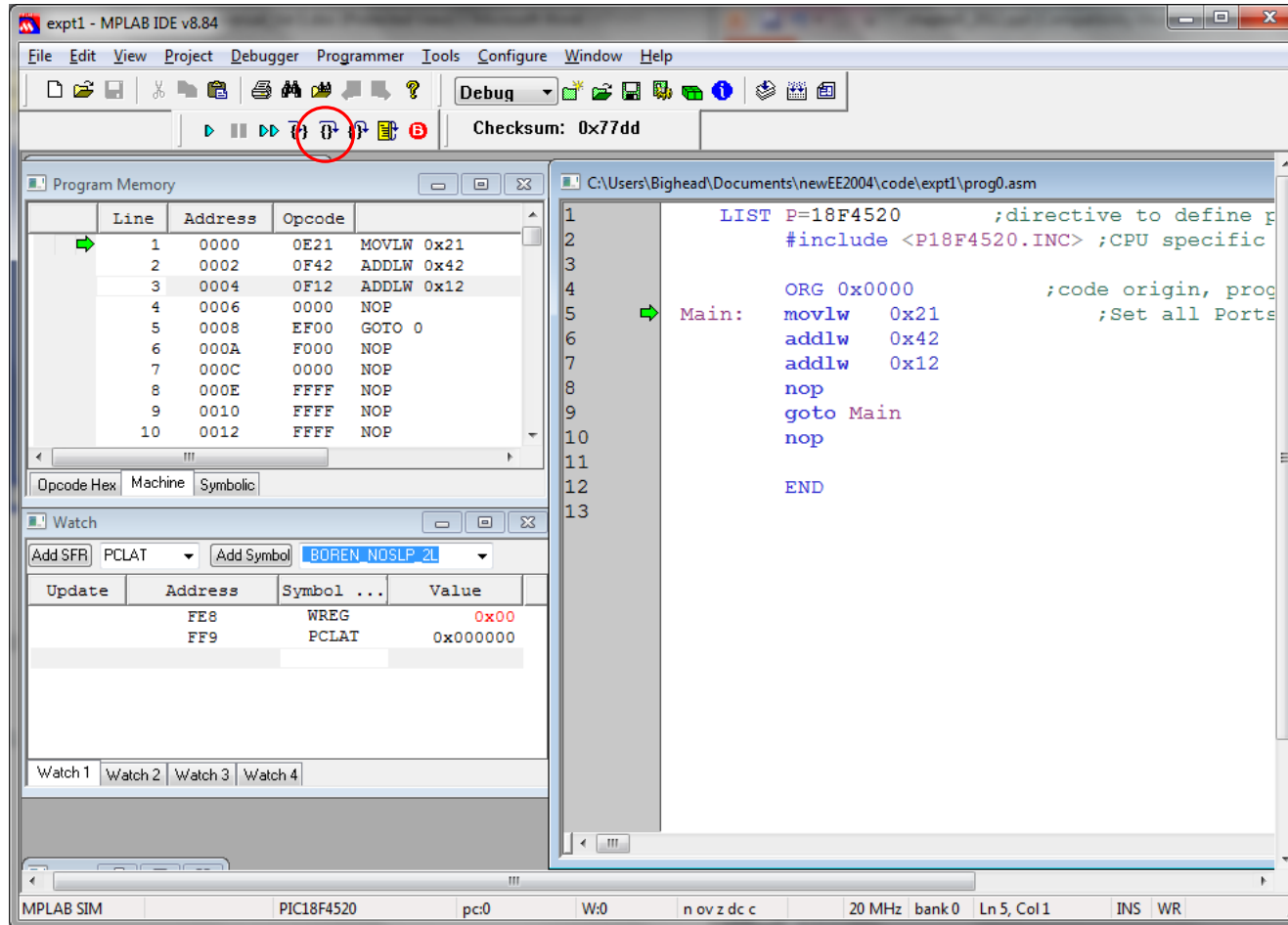


13. Select “PCLAT” and then click “Add SFR” button.  
(Now you can watch PC during the simulation)



14. Click “Step Over” button.

*(You can observe how contents of register change during the execution of the program)*





## Summary

- ◆ von Neumann architecture vs Harvard architecture
- ◆ compare microprocessor and microcontroller
- ◆ application of microcontroller in embedded system
- ◆ overview of PIC18
- ◆ introduction of software development for PIC18