Experiment #10

Introduction to the 8051 Microcontroller

10.0 Objectives:

The objective of this experiment is to learn to use the 8-bit 89C51 microcontroller to implement a simple LED controlling system.

In this experiment, you will do the following:

- Understand the difference between microprocessors and microcontrollers
- Learn about the MCS-51 (8051) microcontrollers – in particular the ATMEL 89C51
- Implement a LED controlling system using the ATMEL 89C51 microcontroller
- Learn to use the Microcontroller/EEPROM programming tool: WINLV

10.1 Equipment, Software, and Components:

- MicroMaster LV48
- WINLV software
- Assembler and conversion utilities (exe2bin, bin2hex)
- AT89C51 microcontroller
- 11.0592 MHz crystal
- Resistors: 510, 8.2K
- Capacitors: 33pF (2), 10pF
- Proto-board (with 5V supply, LED's, switches)

10.2 Introduction:

Microprocessors and Microcontrollers

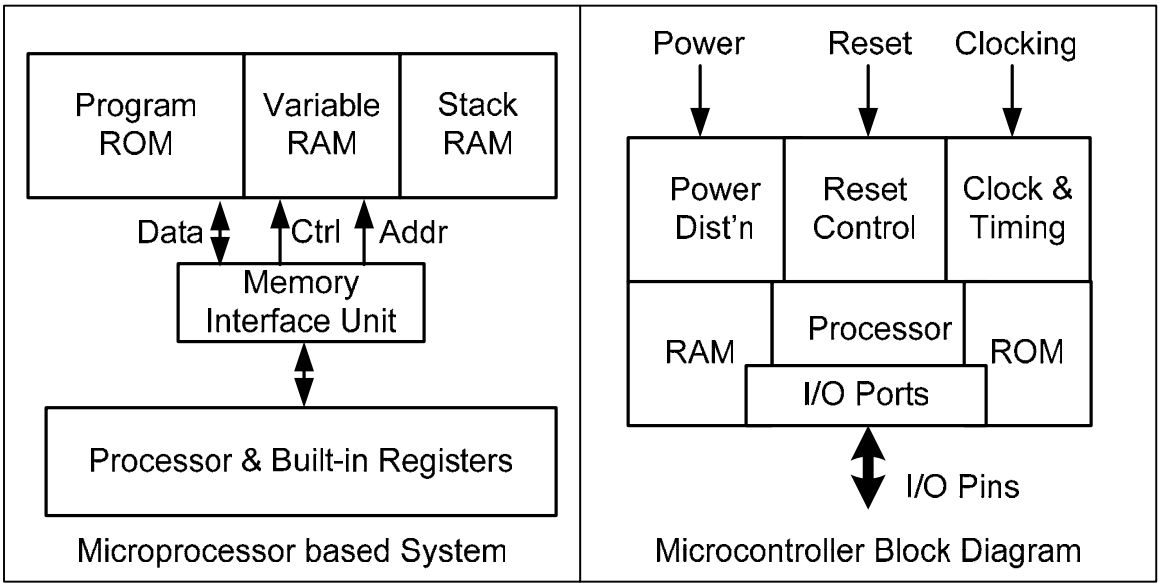
A microprocessor is a **general-purpose** digital computer central processing unit. To make a complete microcomputer, you add memory (ROM and RAM) memory decoders, an oscillator, and a number of I/O devices. The prime use of a microprocessor is to read data, perform extensive calculations on that data, and store the results in a mass storage device or display the results. The design of the microprocessor is driven by the desire to make it as expandable and flexible as possible.

A microcontroller is a true **computer on a chip**. The design incorporates all of the features found in a microprocessor CPU: ALU, PC, SP, and registers, plus ROM, RAM, parallel I/O, serial I/O, counters and a clock circuit – **all in a single IC**. The

microcontroller is a general-purpose device meant to read data, perform limited calculations on that data and control its environment based on those calculations. The prime use of a microcontroller is to control the operations of a machine using a fixed program that is stored in ROM and does not change over the lifetime of the system. The microcontroller is concerned with getting data from and to its own pins; the architecture and instruction set are optimized to handle data in bit and byte size.

Therefore, a microcontroller is a **highly integrated device** which includes, on one chip, all or most of the parts needed to perform an application control function.

Microprocessors vs. Microcontrollers	
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 • expansive • versatile • general-purpose • mostly used in microcomputer systems	**Microcontroller** • CPU, RAM, ROM, I/O and timer, etc. are all on a single chip • fix amount of on-chip ROM, RAM, I/O ports • for applications in which cost, power and space are critical • single-purpose • mostly used in embedded systems



Microcontrollers are frequently found in home appliances (microwave oven, refrigerators, television and VCRs, stereos), computers and computer equipment (laser printers, modems, disk drives), cars (engine control, diagnostics, climate control), environmental control (greenhouse, factory, home), instrumentation, aerospace, and thousands of other uses. In many items, more than one processor can be found.

Microcontrollers come in many varieties. Depending on the power and features that are needed, one might choose a 4, 8, 16, or 32 bit microcontroller. The following table lists some of the commonly used microcontrollers.

4-bit Microcontrollers
Texas Instruments TMS 1000
National COP420
Hitachi HMCS40
Toshiba TLCS47
8-bit Microcontrollers
Intel 8048
Intel 8051
Microchip PIC16C56
National COP820
Motorola 68HC11
Texas Instruments TMS7500
Zilog Z8
16-bit Microcontrollers
Motorola MC68332
Motorola 68HC16
Intel MCS-96 Family of Microcontrollers
National HPC16164
Hitachi H8/532
32-bit Microcontrollers
Intel 80960CA, KA, KB, MC
LR 33000
AMD Am29050
NS 32000

Table: Some of the commonly used microcontrollers

10.2.1 MCS-51 Family of Microcontrollers

The MCS-51 is a family of microcontroller ICs developed, manufactured, and marketed by Intel Corporation. Other IC manufacturers, such as Siemens, AMD, ATMEL, Philips, etc. are licensed “second source” suppliers of devices in the MCS-51 family.

10.2.2 The 8051 Microcontroller

The generic MCS-51 IC is the 8051 8-bit microcontroller, the first device in the family offered commercially. It is the world's most popular microcontroller core, made by many independent manufacturers (truly multi-sourced). There were 126 million 8051s (and variants) shipped in 1993!! Its features are summarized below:

- CPU with Boolean processor
- 4K bytes ROM (factory masked programmed)
- 128 bytes RAM
- Four 8-bit I/O ports
- Two 16-bit timer/counters
- Serial Interface (programmable full-duplex)
- 64K external code memory space
- 64K external data memory space
- Five interrupts (2 priority levels; 2 external)

10.2.3 8051 Flavors

The 8051 has the widest range of variants of any embedded controller on the market. The smallest device is the Atmel 89c1051, a 20 Pin FLASH variant with 2 timers, UART, 20mA. The fastest parts are from Dallas, with performance close to 10 MIPS! The most powerful chip is the Siemens 80C517A, with 32 Bit ALU, 2 UARTS, 2K RAM, PLCC84 package, 8 x 16 Bit PWMs, and other features.

Among the major manufacturers are:

AMD	Enhanced 8051 parts (no longer producing 80x51 parts)
Atmel	FLASH and semi-custom parts, e.g., 89C51
Dallas	Battery backed, program download, and fastest variants
Intel	8051 through 80c51gb / 80c51sl
OKI	80c154, mask parts
Philips	87c748 thru 89c588 - more variants than anyone else
Siemens	80c501 through 80c517a, and SIECO cores

10.2.4 An Architectural Overview of the MCS-51 (8051) Microcontroller

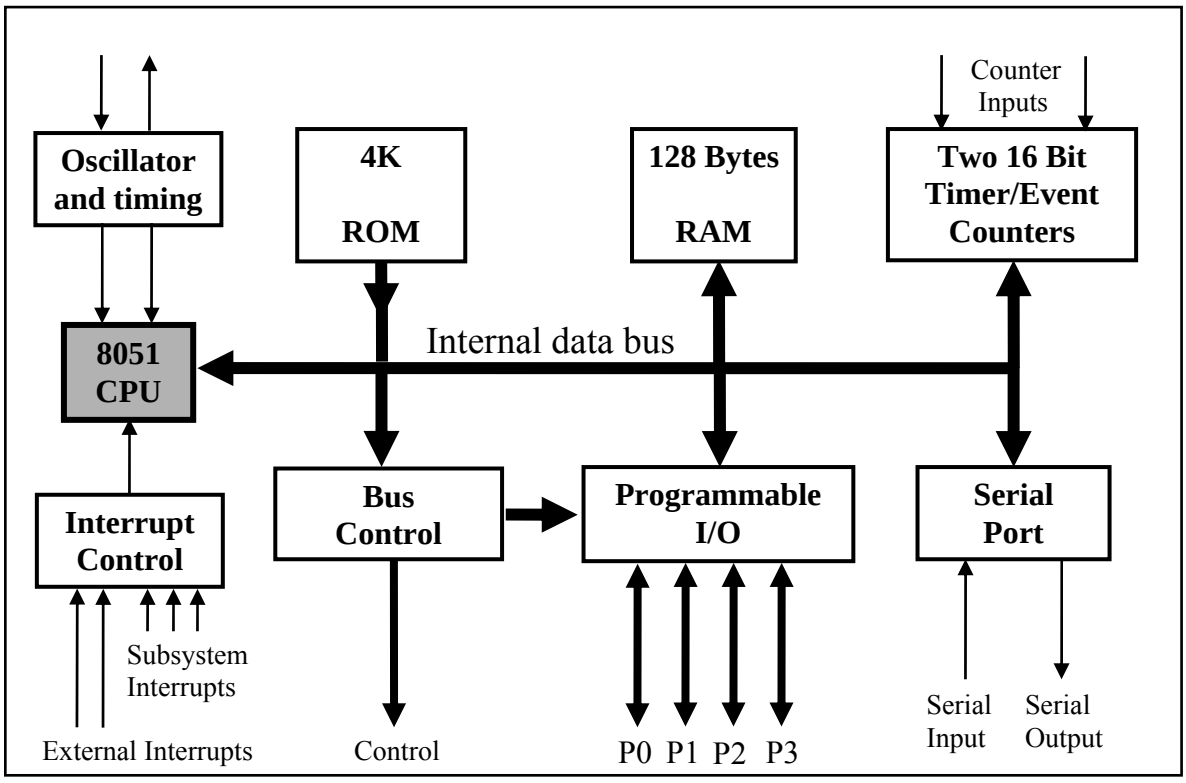
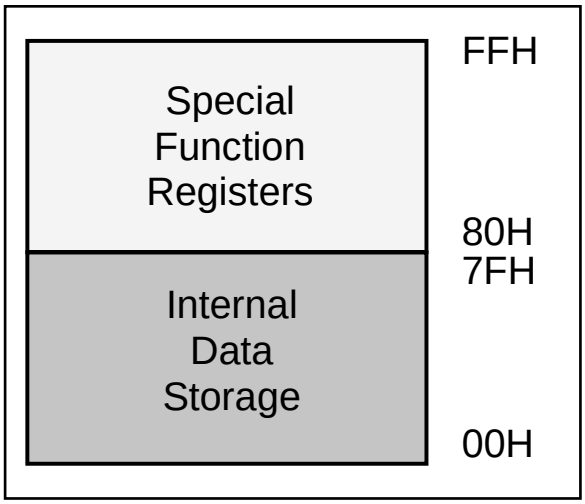


Figure: Block Diagram of the 8051 Core

10.2.5 Data Storage

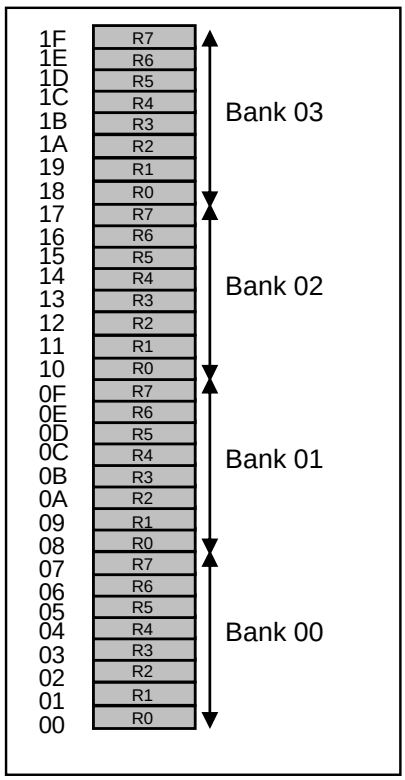
The 8051 has 256 bytes of RAM on-chip. The lower 128 bytes are intended for internal data storage. The upper 128 bytes are the Special Function Registers (SFR). The lower 128 bytes are not to be used as standard RAM. They house the 8051's registers, its default stack area, and other features.



Register Banks

- The lowest 32 bytes of the on-chip RAM form 4 banks of 8 registers each.
- Only one of these banks can be active at any time.
- Bank is chosen by setting 2 bits in PSW
- Default bank (at power up) is bank 0 (locations 00 – 07).
- The 8 registers in any active bank are referred to as R0 through R7.

Given that each register has a specific address; it can be accessed directly using that address even if its bank is not the active one.



10.2.6 Special Function Registers

The upper 128 bytes of the on-chip RAM are used to house special function registers. In reality, only about 25 of these bytes are actually used. The others are reserved for future versions of the 8051.

These are registers associated with important functions in the operation of the MCS-51.

Some of these registers are *bit-addressable* as well as *byte-addressable*. The address of bit 0 of the register will be the same as the address of the register.

- ACC and B registers – 8 bit each
- DPTR : [DPH:DPL] – 16 bit combined
- PC : Program Counter – 16 bits
- Stack pointer SP – 8 bit
- PSW : Program Status Word
- Port Latches
- Serial Data Buffer
- Timer Registers
- Control Registers

See **Appendix B** for a complete list of Special Function Registers and their addresses.

Register A or ACC – Accumulator

This register is commonly used for move operation and arithmetic instructions. It operates in a similar manner to the 8086 accumulator. It is also bit addressable. It can be referred to in several ways:

- Implicitly in op-codes.
- Referred to as ACC (or A) for instructions that allow specifying a register.
- By its SFR address 0E0H.

Register B

It is commonly used as a temporary register. It is also bit addressable.

- Used by two op-codes
  - MUL AB, DIV AB
- B register holds the second operand and will hold part of the result
  - Upper 8 bits of the multiplication result
  - Remainder in case of division.
- Can also be accessed through its SFR address of 0F0H.

DPH and DPL Registers

These are two 8-bit registers which can be combined into a 16-bit DPTR – Data Pointer. The DPTR is used by commands that access external memory.

- Also used for storing 16bit values

```
MOV DPTR, #data16        ; setup DPTR with 16bit ext address
MOVX A, @DPTR            ; copy mem[DPTR] to A
```

- Can be accessed as 2 separate 8-bit registers if needed.
- DPTR is useful for string operations and look up table (LUT) operations.

Port Latches – P0, P1, P2, and P3

These registers specify the value to be output on an output port or the value read from an input port. They are also bit addressable. Each port is connected to an 8-bit register in the SFR.

P0 = 80H, P1 = 90H, P2 = A0H, P3 = B0H

- First bit has the same address as the register.
- Example: P2 has address A0H in the SFR, so
  - P2.7 or A7H refer to the same bit.
- All ports are configured for output at reset.

PSW – Program Status Word

Program Status Word is a bit addressable 8-bit register that has all the status flags.

CY	AC	F0	RS1	RS2	OV	-	P

Symbol	Position	Function
CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag. For BCD Operations
F0	PSW.5	Flag 0. Available to the user for general purposes.
RS1	PSW.4	Register bank select bits. Set by software to determine which register bank is being used.
RS2	PSW.3	
OV	PSW.2	Overflow Flag
-	PSW.1	Not used
P	PSW.0	Parity Flag. Even Parity.

10.2.7 8051 Instructions

The 8051 has 255 instructions. Every 8-bit op-code from 00 to FF is used except for A5. The instructions are grouped into 5 groups:

- Arithmetic
- Logic
- Data Transfer
- Boolean
- Branching

The following table lists some of the commonly used instructions. See **Appendix A** for a complete list of the 8051 instructions.

Instruction	Description	Execution Cycle
ACALL sub1	Call a subroutine labeled sub1	2
CJNE a, b, addr	Compare a and b, if they are not equal then jump to addr	2
CLR x	Set the bit x to 0	1
DJNZ a, addr	Decrease a by 1, then check if a = 0, then jump to addr	2
INC a	Increase a by 1	1-2
MOV a, b	Move the content in b to a	1-2
MOVC a, addr	Move the content stored in address addr to a	2
RET	Return to main program from a subroutine	2
RETI	Return to main program from an interrupt subroutine	2
SETB x	Set the bit x to 1	1
SJMP addr	Jump to the addr	2
XRL a, b	Perform a XOR b (XOR: Exclusive Or)	1-2

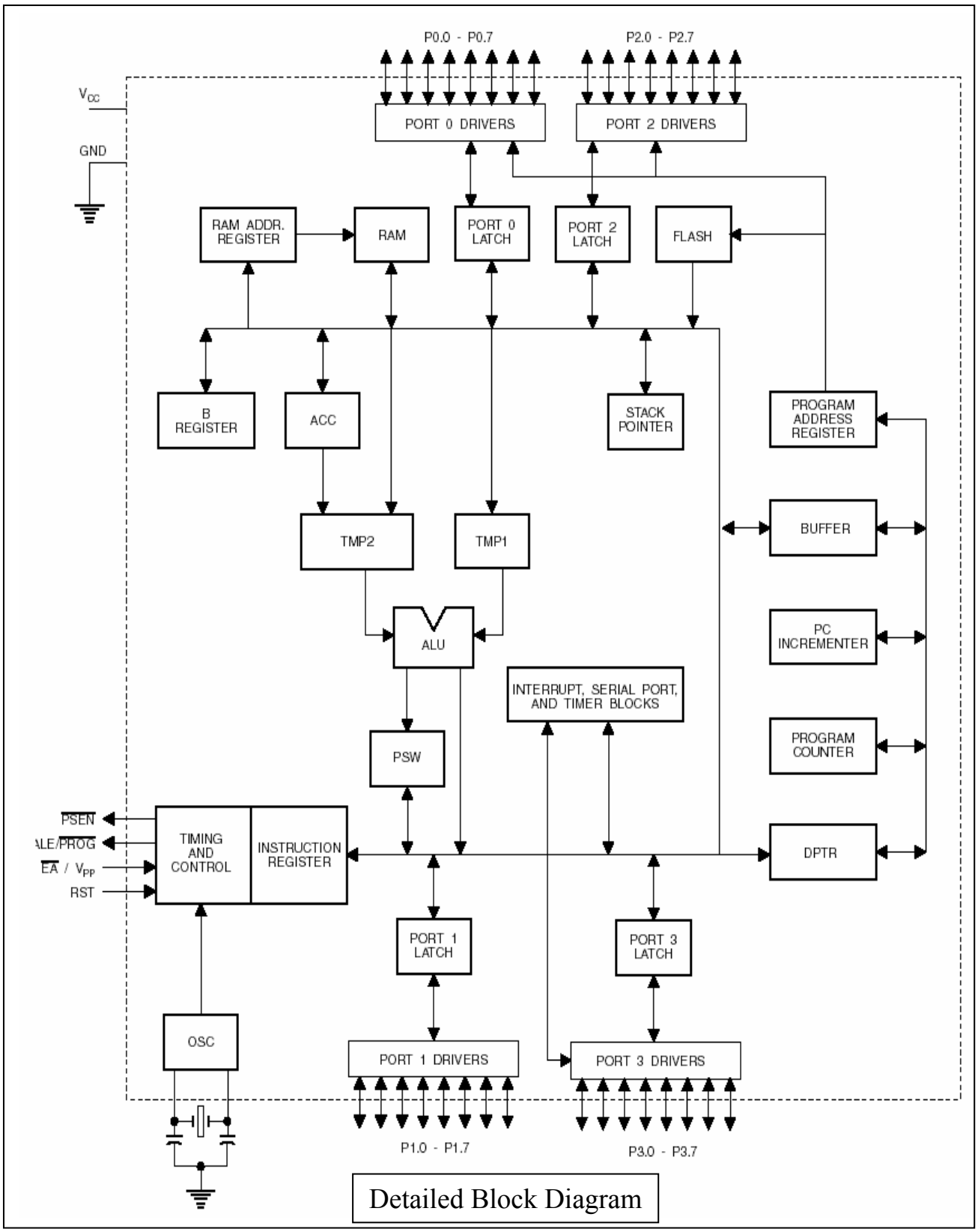
10.2.8 ATMEL 89C51 Microcontroller

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of **Flash Programmable and Erasable Read Only Memory** (PEROM). The device is manufactured by Atmel and is compatible with the industry-standard MCS-51 instruction set and pin-out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Features

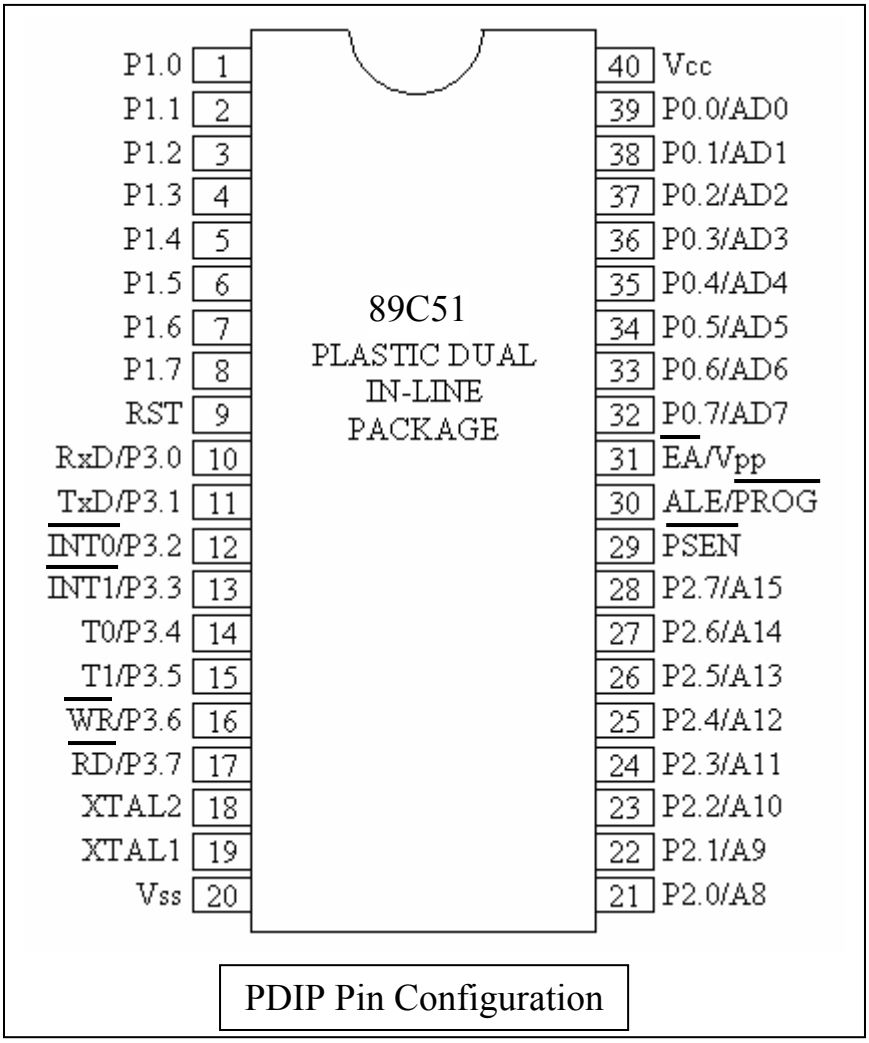
- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

A detailed Block Diagram of the AT89C51 microcontroller is shown in the figure below.



Detailed Block Diagram

The AT89C51 comes in a 40 pin package. The Pin Configuration of the AT89C51 microcontroller is shown in the diagram below.



PDIP Pin Configuration

32 pins are used for the 4 ports - **P0, P1, P2, and P3**
1 pin each for **V_{CC}** and **V_{SS}**
1 pin for the **ALE (**Address Latch Enable)
1 pin for **EA/V_{PP}** • **EA** - External Address • **V_{PP}** - Program Voltage for EPROM based versions of the 8051
1 pin each for **XTAL1** and **XTAL2** - Connections for clock crystal
1 pin for **PSEN** - "Program Store Enable" • Read signal for external program memory
1 pin for **RST** - Reset

10.3 Pre-lab:

1. Review all the above sections of this experiment and **Appendix A, B, and C.**
2. Look for information on the 8051 microcontroller in the library and on the internet.

10.4 Lab Work:

In this experiment, we will implement a simple LED control system on the 89C51 microcontroller. This implementation involved three steps:

1. Software programming 2. Hardware connections 3. Microcontroller programming

Procedure: **Software**

1. Use an editor to write the program shown below. Name your file as ***led_blink.asm***.

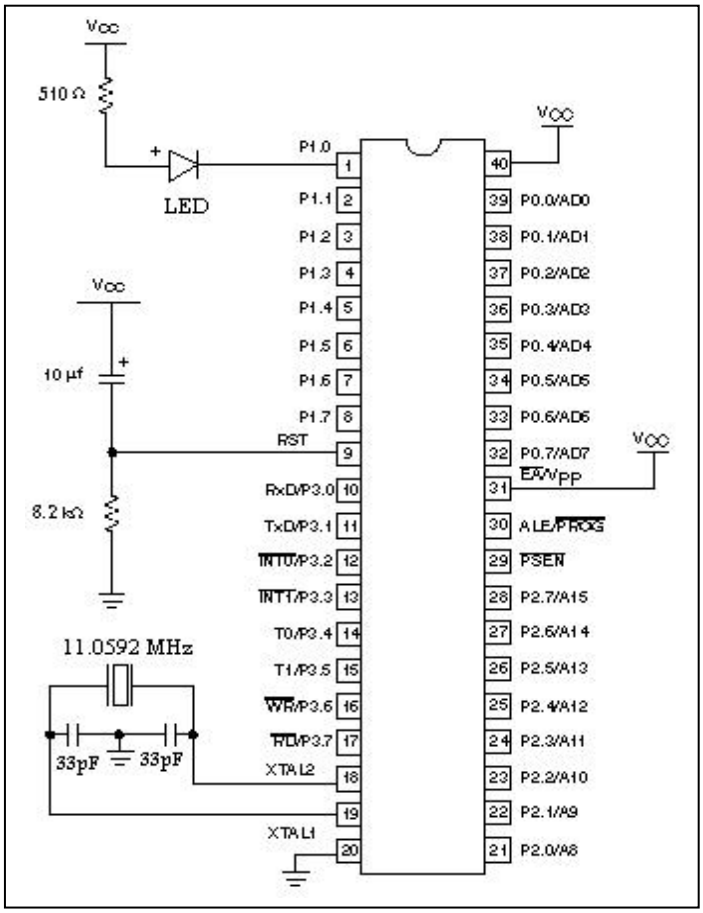
```
; This program blinks LEDs every half second

ORG 00H                     ; After reset, start fetching instructions from 00H

        MOV A, #55H         ; load 0101 0101 in A
AGAIN:
        MOV P1, A           ; move data to Port A

        ACALL DELAY         ; call delay routine

        CPL A               ; invert A to 1010 1010
        SJMP AGAIN          ; repeat blinking

DELAY:                      ; this routine implements a delay 0f 500ms
        MOV R4, #05         ; move 5 in R4
OUTER2: MOV R3, #200        ; move 200 in R3
OUTER1: MOV R2, #0255       ; move 255 in R2
INNER:  DJNZ R2, INNER      ; decrement R2 until 0
        DJNZ R3, OUTER1     ; decrement R3 until 0
        DJNZ R4, OUTER2     ; decrement R4 until 0
        RET

END
```

2. Assemble and link your program using the TASM assembler to produce ***led_blink.exe***.

3. Generate the HEX file ***led_blink.hex*** for your program.

Procedure: **Hardware**

1. Turn OFF the power supply to your board.
2. Obtain the necessary components from your lab instructor and connect the circuit as shown in the schematic below:



3. Connect the rest of the pins of Port 1 to an LED each.
   Note 1: Only P1.0 (Pin 0 of Port 1) is shown connected to an LED.
   Note 2: You may be directly able to connect to LED's on the board provided without using any resistor with the LED's. Check with your lab instructor.

4. Make sure all your connections are properly done and turn ON the power supply.

5. Observe that no LED is blinking.

Procedure: **Microcontroller Programming**

Step 1: Run WinLV
Launch the WinLV program. See **Appendix C** - An Introduction to WinLV.

Note: During start-up, the red light of the programmer will go ON indicating that the software has successfully found the programmer. If the software fails it will display a prompt to enter a demo-mode. If this is the case, double check the cable connected to the parallel port and the power supply of the programmer.

Step 2: Select Device (ATMEL 89C51 microcontroller)
Choose the device that you are using by selecting it from the ***Select Device*** window. To open the ***Select Device*** window – click the left hand mouse button on ***Programmer*** from the top menu bar then click on ***Select Device***. Use the mouse to browse through the Parts Database and select a device from the Programmable Parts section then click on Accept to confirm your choice.

Step 3: Load File into Buffer
Select the file you would like to program into the Microcontroller following these steps:-

- Select the ***File*** menu and then ***Open…***
  Select which file you want to load into the buffer OR type the file name into the ***File Name*** box. Locate the ***led_blink.hex*** file generated from the assembly code.
- If any files that you expecting to see aren't there try using the ***Files of Type*** drop down menu to select which file type is displayed in the main window.
- Use the ***Open as*** drop down menu to select which Format the file will be opened as and loaded into the buffer. Make sure that the value of the field "Open as" is : HEX – Intel" or " HEX - auto recognition ".
- Set the Defaults to pre fill the buffer with 0xFF so that all empty address locations are filled with FF.

Step 4: Place Device in Programmer
Place the device you've selected to program into the ZIF socket on the programmer. Make sure that the device is aligned with the bottom of the ZIF socket.

Step 5: Programme the Device
Follow these final steps to finish programming the device:

- Select ***Operations*** from the ***Programmer*** menu to view the “Operations for Device” dialogue window.
- Click on the ***Programme*** button in the ***Operations*** dialogue window.
- The device in the socket will now be erased, programmed and verified.

Finally, disconnect the IC from the programmer and reconnect it to the circuit. Reset the microcontroller and check if your program is working correctly.