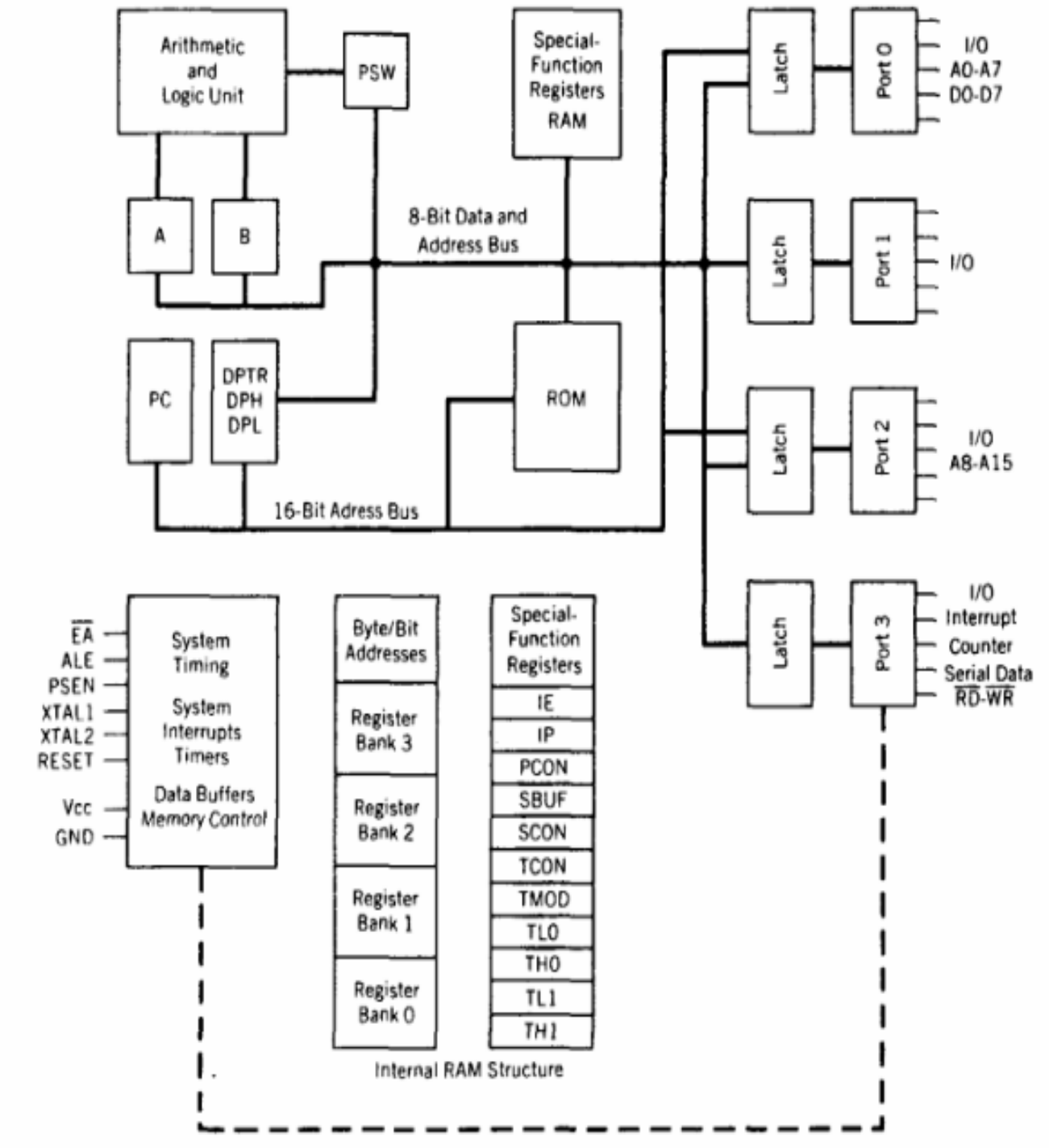
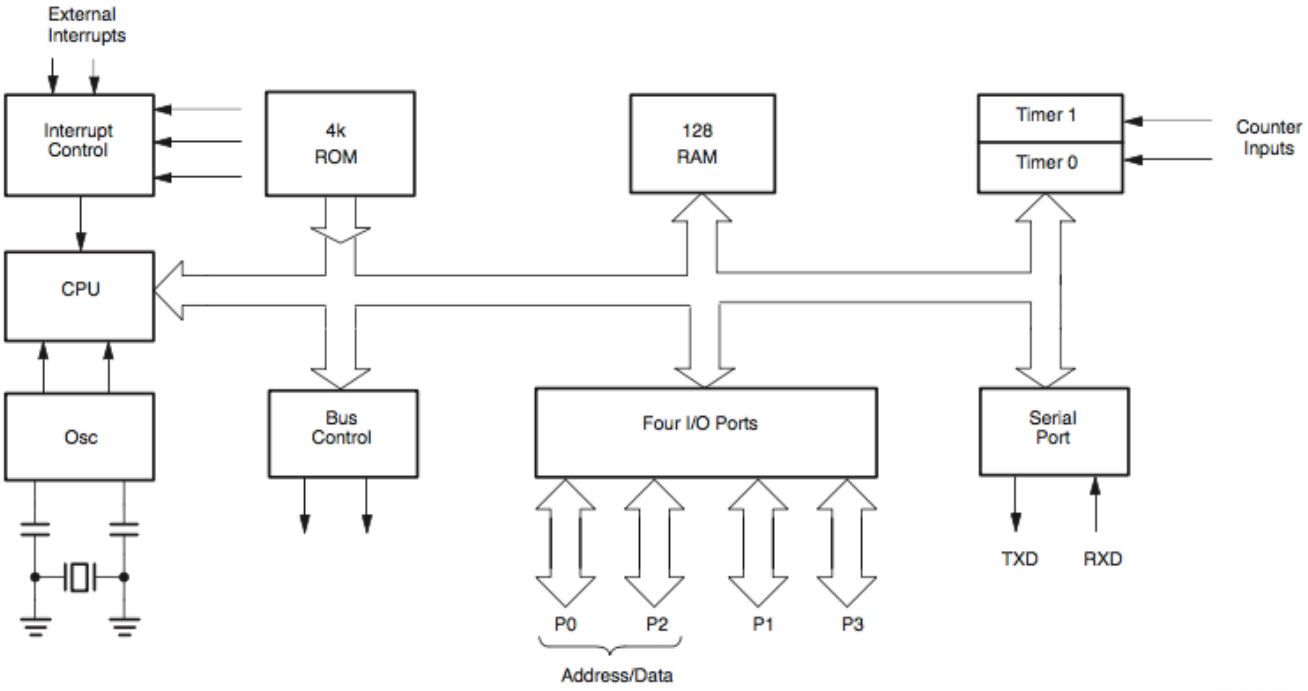
8051 BLOCK DIAGRAM



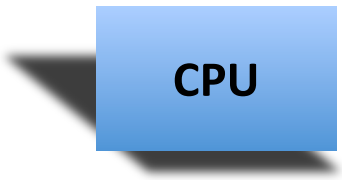
Alternate diagram …

ALTERNATE DIAGRAM FOR 8051 ARCHITECTURE / BLOCK DIAGRAM



8051 is a microcontroller. This means it has an internal processor, internal memory and an I/O section. The architecture of 8051 is thus divided into three main sections:

- The CPU
- Internal Memory
- I/O components.



8051 has an 8 bit CPU.
This is where all 8-bot arithmetic and logic operations are performed.
It has the following components.

ALU – ARITHMETIC LOGIC UNIT

It performs **8-bit arithmetic and logic operations**.
It can also perform some bit operations.

```
Example:
ADD A, R0       ; Adds contents of A register and R0 register and stores the result in A register.
ANL A, R0       ; Logically ANDs contents of A register and R0 register and stores the result in A register.
CPL P0.0        ; Complements the value of P0.0 pin.
```

A – REGISTER (ACCUMULATOR)

It is an **8-bit register**.
In most arithmetic and logic operations, **A register hold the first operand and also gets the result of the operation**.
Moreover, it is the only register to be used for **data transfers** to and from **external memory**.

```
Example:
ADD A, R1       ; Adds contents of A register and R1 register and stores the result in A register.
MOVX A, @DPTR   ; A gets the data from External RAM location pointed by DPTR
```

B – REGISTER

It is an **8-bit register**.
It is dedicated for **Multiplication and Division**.
It can also be used in other operations.

```
Example:
MUL AB          ; Multiplies contents of A and B registers. Stores 16-bit result in B and A registers.
DIV AB          ; Divides contents of A by those of B. Stores quotient in A and remainder in B.
```

PC – PROGRAM COUNTER

It is an **16-bit register**.
It holds **address of the next instruction** in program memory (ROM).
PC gets automatically **incremented** as soon as any **instruction is fetched**.
That's what makes the program move ahead in a **sequential manner**.
In the case of a **branch**, a **new address is loaded into PC**.

DPTR – DATA POINTER

It is an **16-bit register**.
It holds **address data** in data memory (RAM).
DPTR is divided into two registers DPH (higher byte) and DPL (lower byte).
It is typically used by the programmer **to transfer data from External RAM**.
It can also be used as a pointer to a **look up table** in ROM, using **Indexed addressing mode**.

Example:
MOVX A, @DPTR ; *A gets the data from External RAM location pointed by DPTR*
MOVC A, @A+DPTR ; *A gets the data from ROM location pointed by A + DPTR*

SP – STACK POINTER

It is an **8-bit register**.
It contains **address of the top of stack**.
The Stack is present in the Internal RAM.
Internal RAM has 8-bit addresses from 00H… 7FH.
Hence SP is an 8-bit register.
It is affected during Push and Pop operations.
During a **Push**, SP gets **incremented**.
During a **Pop**, SP gets **decremented**.

PSW – PROGRAM STATUS WORD

It is an **8-bit register**.
It is also called the **"Flag register"**, as it mainly contains the status flags.
These flags indicate **status of the current result**.
They are **changed by the ALU after every arithmetic or logic operation**.
The flags can also be **changed by the programmer**.
PSW is a **bit addressable** register.
Each bit can be individually set or reset by the programmer.
The bits can be referred to by their bit numbers (**PSW.4**) or by their name (**RS1**).

Example:
SETB PSW.3 ; *Makes PSW.3 ← 1*
CLR PSW.4 ; *Makes PSW.4 ← 0*