

8051 Architecture and Addressing Modes

8051 assembly language is one of the parts of this course. But first we will talk about Intel's 8051 microcontroller architecture and addressing modes in order to understand its assembly language better.

The references that you can use for 8051 architecture, addressing modes and assembly language are on the web site. These are 8051 architecture datasheet (<http://www.ece.orst.edu/~sllu/471/3081.pdf>), 8051 programmer's guide and instruction set datasheet (<http://www.ece.orst.edu/~sllu/471/3082.pdf>). Moreover, the instruction set summary is at <http://www.ece.orst.edu/~sllu/471/doc513.pdf>. 8051 Hardware datasheet is also at the web site (<http://www.ece.orst.edu/~sllu/471/3083.pdf>).

- 8051 is the core member of MCS-51 microcontroller family.
- Single chip microcomputer
 - Embedded RAM and ROM/EPROM
 - I/O ports
 - Parallel ports (8-bit is transferred together)
 - Serial port (bit-by-bit)
- It is designed by Intel but has many variants from different manufacturer
- 8-bit machine, because “data width” = “Acc. Width” = 8-bit.
- Most of the other registers are also 8-bit. PC, Data Pointer (DPTR), timer registers are 16-bit but they are made of two 8-bit registers.

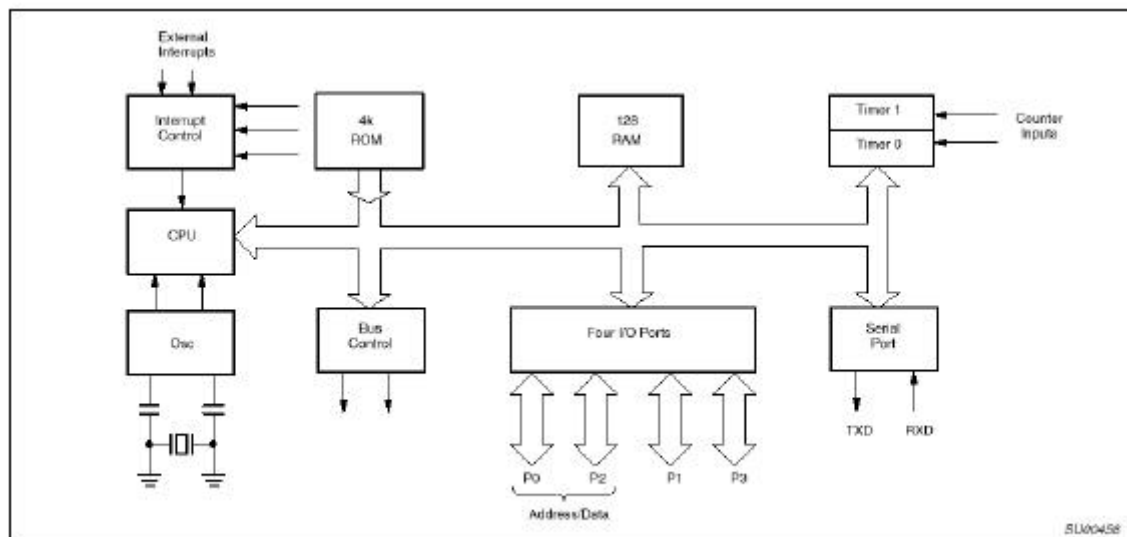


Figure 1. 80C51 Block Diagram

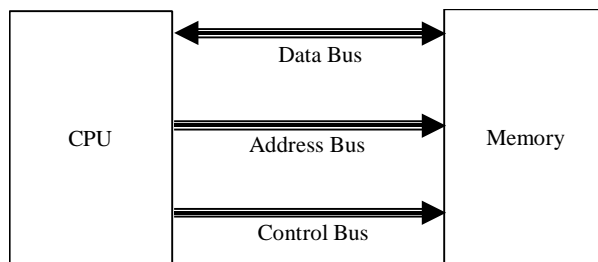
8051 block diagram is given above. Special Function Registers (SFRs), Arithmetic and Logic Unit, Control Unit are assumed to be the parts of CPU.

- There is embedded 4K (4096 bytes) of ROM/EPROM on-chip
- There is a total of 128 bytes of RAM on-chip. 32 of them are general purpose registers (no specific job), 96 bytes consist of internal data memory.
- There are 4 parallel I/O ports each 8-bit. These ports are used to communicate with outside world 8-bit at a time.
- There are 2 timers/counters each 16-bit. They can be programmed to be used as either as timer to count internal clock cycles or as counters to count external input cycles. We'll not deal with the details of them in this course.
- There is 1 programmable, bidirectional (both receive and transmit), asynchronous serial port. This port is also used for data transfer but one bit at a time. We'll not deal with the details of serial port in this course either.
- Interrupt mechanism allows the computers to respond asynchronous external and internal events without waiting for them in a dedicated manner. In 8051 there is an "interrupt control unit" and 5 interrupt sources. Two of them are from external lines (External Interrupts), two from timers/counters as the overflow bits, one from the serial port to check the transmission and reception of data. We will deal with only the external interrupts in this course.
- In CPU, besides the ALU and control unit, there are 21 addressable SFRs and some non-addressable registers (like PC, IR, Address Registers). Addressable SFRs can be used in direct addressing mode in instructions, while the non-addressable registers cannot.

Memory Organization

Generally, a CPU has three different connections to the memory unit. These are

- Data lines (bus) (to read/write data – bidirectional)
- Address lines (bus) (to send the address of the memory location – unidirectional)
- Control Lines (bus) (to send read/write signals to the memory unit – unidirectional)



- The number of bits in data bus is the *data width* of the CPU and the memory unit. In 8051, data width is 8-bit.
- The maximum number of bits in the address bus is the CPU's *address space*. However, there can be less number of bits there, if the actual memory used by the CPU is less than the maximum.

In 8051, data memory and the program memory are different. This is a characteristic of the "Harvard Architecture". Instructions and programs are stored in *program memory*. Data used in these instructions is stored in data memory. There are different control

signals for the program and the data memories. There are also different address spaces. But the data and the address lines are multiplexed.

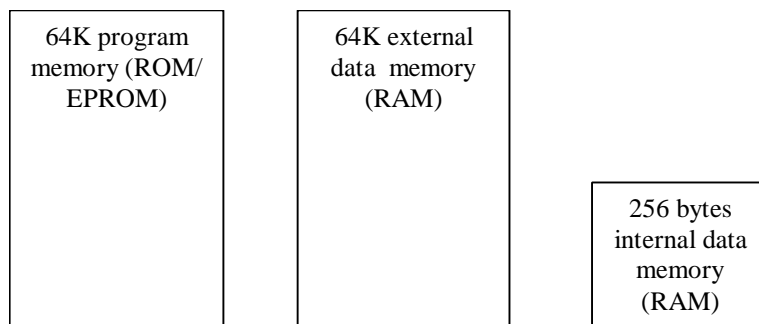
Program memory: In 8051 program memory can be external (not on 8051 chip), internal (on the 8051 chip, 4K capacity) or both. The total address space is 16-bit. Therefore, we can have up to $2^{16} = 64\text{K}$ program memory. EPROMs/ROMs are used as the program memory.

Data memory: There are two different data memories in 8051. One of the internal and the other is external. Internal data memory is on the chip. External data memory is a separate chip. There are different address spaces, different control signals and especially different instructions for them. In this course, we'll talk about the internal data memory instructions and in the labs, we'll assume that no external data memory connected. RAMs are used as data memory.

The address space of the internal data memory is 8-bit. Therefore, we can have $2^8 = 256$ addressable bytes. But, not all of this space is used in 8051. Using this address space, we address 21 addressable SFRs + 32 general purpose registers + 96 bytes of RAM = 149 addresses.

External data memory address space is 16-bit. Therefore, we can have up to $2^{16} = 64\text{K}$ addresses (bytes) there.

The overall address space of 8051 is as the following



Internal Memory Organization

As we discussed earlier, internal data memory (internal RAM) address space is 8-bit. Therefore, we have 256 addressable bytes here. Moreover, there are some addressable “bits” in internal data memory, too. The “bit address space” is separate from the byte address space. The bit address space is also 8-bit, so we have 256 addressable bits in the internal data memory.

Internal data memory organization is shown below (Courtesy of Prof. Shih-Lien Lu).

| bit-addressable | | | | | | | | | | | | | | | | |
|----------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| banks00 | | 01 | | 10 | | 11 | | | | | | | | | | |
| R7 | R7 | R7 | R7 | 0F | 17 | 1E | 27 | 2E | 37 | 3E | 47 | 4E | 57 | 5E | 67 | 7E |
| R6 | R6 | R6 | R6 | 0E | 16 | 1B | 26 | 2B | 36 | 3B | 46 | 4B | 56 | 5B | 66 | 7B |
| R5 | R5 | R5 | R5 | 0D | 15 | 1D | 25 | 2D | 35 | 3D | 45 | 4D | 55 | 5D | 65 | 7D |
| R4 | R4 | R4 | R4 | 0C | 14 | 1C | 24 | 2C | 34 | 3C | 44 | 4C | 54 | 5C | 64 | 7C |
| R3 | R3 | R3 | R3 | 0B | 13 | 1B | 23 | 2B | 33 | 3B | 43 | 4B | 53 | 5B | 63 | 7B |
| R2 | R2 | R2 | R2 | 0A | 12 | 1A | 22 | 2A | 32 | 3A | 42 | 4A | 52 | 5A | 62 | 7A |
| R1 | R1 | R1 | R1 | 09 | 11 | 19 | 21 | 29 | 31 | 39 | 41 | 49 | 51 | 59 | 61 | 79 |
| R0 | R0 | R0 | R0 | 08 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 68 | 78 |
| Byte00-addr 07 08-1F 10-1F R0-R7 | | | | | | | | | | | | | | | | |
| Internal RAM | | | | | | | | | | | | | | | | |
| General Purpose RAM (80 Bytes) | | | | | | | | | | | | | | | | |
| 30-7F | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|--------------------|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Byte00-addr 80 | | | | | | | | | | | | | | | |
| P0 | S7 | | | | | | | | | | | | | | |
| SP | S6 | | | | | | | | | | | | | | |
| DPL | S5 | | | | | | | | | | | | | | |
| DPH | S4 | | | | | | | | | | | | | | |
| PCON | S3 | | | | | | | | | | | | | | |
| TCON | S2 | | | | | | | | | | | | | | |
| TMOD | S1 | | | | | | | | | | | | | | |
| TL0 | S0 | | | | | | | | | | | | | | |
| TH0 | S7 | | | | | | | | | | | | | | |
| TH1 | S6 | | | | | | | | | | | | | | |
| P1 | S5 | | | | | | | | | | | | | | |
| SCON | S4 | | | | | | | | | | | | | | |
| SBUF | S3 | | | | | | | | | | | | | | |
| P2 | S2 | | | | | | | | | | | | | | |
| IE | S1 | | | | | | | | | | | | | | |
| P3 | S0 | | | | | | | | | | | | | | |
| IP | D7 | | | | | | | | | | | | | | |
| PSW | D6 | | | | | | | | | | | | | | |
| ACC | D5 | | | | | | | | | | | | | | |
| B | D4 | | | | | | | | | | | | | | |
| SPECIAL FUNC. REG. | | | | | | | | | | | | | | | |

Lower 128 bytes (upper part of the figure above):

- Byte addresses are 00H – 7FH.
- Accessible via direct and indirect addressing
- There are 32 general purpose registers in 4 banks (8 registers per bank, R0 – R7). The addresses of these locations are between (and including) 00H – 1FH. Only one bank is active at a time and when you specify a register (R0, R1, R2, ... , R7), you refer to the register in the active bank. Active bank selection is done by using two bits of the Program Status Word (PSW). Default bank is 00.

3 bits are enough for referring a general purpose register in an instruction, since we have $2^3 = 8$ active registers at a time.

- There is a bit addressable memory (16 bytes / 128 bits). Byte addresses are between (and including) 20H – 2FH. Bit addresses are between (and including) 00H – 7FH. Bit addresses are shown in cells in the figure above.

Although bit and byte addresses overlap, this does not cause a problem, since there are different instructions for the bits and bytes. For example, “MOV A, 21H” is a byte move instruction, “MOV C, 6FH” is a bit move instruction.

- There is a general purpose RAM (80 bytes). Byte addresses are between (and including) 30H – 7FH. This part of the memory is not bit addressable.

Upper 128 bytes (lower part of the figure above):

- Byte addresses are 80H – FFH.
- Accessible via only direct addressing
- SFRs exist in this part of the internal data memory and there are only 21 of them. 128 - 21 = 107 locations are not used.
- Some SFRs (the ones whose hexadecimal bit addresses end with 0 or 8) are bit addressable.
- No need to memorize SFR addresses, you can use their acronyms in programs as the addresses. For example, in order to move the content of the Port 1 register (P1) to accumulator, you don't need to use the address of the P1 register, you can simply have “MOV A, P1”. Assembler puts the address of P1 (90H) there automatically while assembling the program.
- In order to specify a bit of a SFR, you can use a point (.) between the register and the bit number. For example, in order to complement the 3rd bit of the register B, you can write “CPL B.3”. Assembler puts the address of B.3 (F3H) there automatically while assembling the program. Do not forget that the LSB of a register is bit 0, MSB is bit 7.

SFRs:

P0, P1, P2, P3: Parallel port registers, each 8-bit. They contain the data transmitted/received from/to ports.

P0 and P2 are used as address/data bus for the external program and data memories. But we assume that no external memory is used in labs, so this rule does not apply to the labs. In labs, you can use P0, P2 as regular I/O ports.

P3 carries some special signals for serial port, timers/counters, external data memory access and external interrupt inputs. Except external interrupts, other signals will not be our concern.

P1 is a pure I/O port.

IE: Interrupt Enable register. The bits in that register are used to enable/disable interrupts.

IP: Interrupt Priority register. There are two priority levels for 8051 registers. The bits in that register are used to determine the priority level of each interrupt.

Acc: Accumulator

B: B register. *B* does not stand for anything there. Most probably it is named as B, since accumulator is named as A and B is the next letter after it ☺. Anyway, B register is used for division and multiplication.

PSW: Program Status Word. PSW contains several status bits that reflect the current state of CPU. Let's analyze the PSW in more detail.

| | | |
|-------|------------------------------|--|
| Bit 0 | Parity bit (P) | Parity of accumulator, automatically set/cleared. It is set (becomes 1) if the total number of 1's in acc. is an odd number. It is cleared (becomes 0) if the total number of 1's in acc. is an even number. E.g. if the acc is 5, then P becomes 0, since 5 is 00000101 in binary and there two "1"s in it. |
| Bit 1 | | User defined bit/flag. No specific usage. |
| Bit 2 | Overflow (OV) | Set by the arithmetic operations in case of an overflow |
| Bit 3 | Register Bank Select 0 (RS0) | Low bit of the register bank select |
| Bit 4 | Register Bank Select 1 (RS1) | High bit of the register bank select |
| Bit 5 | | User defined bit/flag. No specific usage. |
| Bit 6 | Auxiliary Carry (AC) | Used for BCD addition/conversion. Carry out of bit 3 in addition |
| Bit 7 | Carry (C, CY) | Carry out of sign bit in addition, subtraction. It is also used as accumulator in bitwise operations |

SP: Stack Pointer. It is a 8-bit register that keeps the top of the stack.

DPTR: Data Pointer. It is a 16-bit register and made of two 8-bit registers DPL (data pointer low) and DPH (data pointer high). DPTR is used as the address register for external data memory access (not useful in labs, since we don't have external data memory in labs).

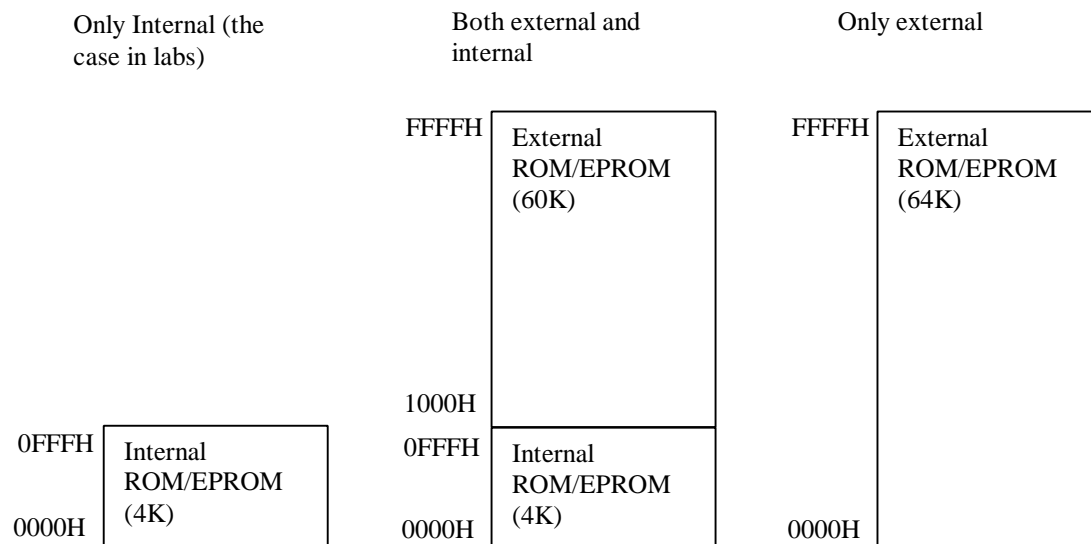
SCON, SBUF: Serial port control and serial port buffer registers. We'll not cover serial ports in this course.

TCON, TMOD, TH0, TH1, TL0, TL1: Timer Control, Timer Mode registers and the two 16-bit timers/counters (each is made of two 8-bit registers). We'll not cover timers in this course.

Program Memory

As discussed earlier, program memory address space is 16-bit and we can have up to 64K program memory.

Program memory can be external, internal or both. These alternatives are given below together with address boundaries. Internal program memory is limited to 4K. The address space is shared between external and internal program memory.



8051 Addressing Modes

Direct Addressing

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal data memory (including SFRs) can be directly addressed.

Examples:

```
MOV A,066H ;move the content of internal data memory address 66H to acc
ADD A,B ;add the content of the B register to acc
```

In the latter example the address of the B register (F0) register is actually used in the encoded form.

Indirect Addressing

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external data memory can be indirectly addressed (SFRs are not indirectly addressed). The address register for 8-bit addresses can only be R0 or R1 of the selected bank, or the Stack Pointer (no other registers or memory location).

“@” symbol is used before the register to specify the indirect addressing mode in instructions.

The address register for 16-bit addresses can only be the 16-bit “data pointer” register, DPTR (this is not the case in labs).

Example:

```
MOV A,@R0 ;move the content of internal data memory location whose
           address is in register R0
```

Register-Specific Instructions

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator specific opcodes.

Example:

```
CPL A ; Complement accumulator
```

The encoded form is 11110100 and does not contain the address of accumulator.

Register Instructions (Register Addressing)

The register banks, containing registers R0 through R7, can be accessed by certain instructions that carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed.

Example:

ADD A, R5 ; add the content of R5 to accumulator

Encoded form is 00101101. The least significant 3 bits (101 = 5) specifies R5.

Immediate Constants (Immediate Addressing)

Actually this is not an addressing mode because no address is specified in the instructions. Here a constant value is specified in the instruction and that value follows the opcode in the encoded form.

“#” symbol precedes the constant values in the assembly instructions. The constants can be decimal numbers, hexadecimal numbers, binary numbers or characters.

Hexadecimal numbers are written with “H” suffix (e.g. 3AH). Moreover, an hexadecimal number must start with a digit. If not, put a leading 0. For example, if the hexadecimal number that you want to specify is D3H, then you have to write it in a program as 0D3H.

Binary numbers are written with a “B” suffix (e.g. 01000101B).

Decimal numbers do not require any suffix.

Characters must be written between double quotation marks (e.g. "Z"). Actually, the ASCII code of the character is processed here.

Example:

MOV A, #100 ;loads the Accumulator with the decimal number 100. The
same number could be specified in hex digits as 64H.