8051 Addressing Mode and Instruction Set
8051 Instruction Set

- **Addressing Modes**
  - Register addressing
  - Direct addressing
  - Indirect addressing
  - Immediate constant addressing
  - Relative addressing
  - Absolute addressing
  - Long addressing
  - Indexed addressing

- **Instruction Types**
  - Arithmetic operations
  - Logical operations
  - Data transfer instructions
  - Boolean variable instructions
  - Program branching instructions
Introduction

- A computer instruction is made up of an operation code (op-code) followed by either zero, one or two bytes of operands.
- The op-code identifies the type of operation to be performed while the operands identify the source and destination of the data.
- The operand can be:
  - The data value itself
  - A CPU register
  - A memory location
  - An I/O port
- If the instruction is associated with more than one operand, the format is always:
  \[ \text{Instruction} \quad \text{Destination, Source} \]
The memory organization of C8051F020 is similar to that of a standard 8051.

Program and data memory share the same address space but are accessed via different instruction types.
Internal Data Memory

DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE

Byte Address | Bit Address
-------------|--------------
0x7F         | General Purpose RAM
30           |
2F 7F 7E 7D 7C 7B 7A 79 78 |
2E 77 76 75 74 73 72 71 70 |
2D 6F 6E 6D 6C 6B 6A 69 68 |
2C 67 66 65 64 63 62 61 60 |
2B 5F 5E 5D 5C 5B 5A 59 58 |
2A 57 56 55 54 53 52 51 50 |
29 4F 4E 4D 4C 4B 4A 49 48 |
28 47 46 45 44 43 42 41 40 |
27 3F 3E 3D 3C 3B 3A 39 38 |
26 37 36 35 34 33 32 31 30 |
25 2F 2E 2D 2C 2B 2A 29 28 |
24 27 26 25 24 23 22 21 20 |
23 1F 1E 1D 1C 1B 1A 19 18 |
22 17 16 15 14 13 12 11 10 |
21 0F 0E 0D 0C 0B 0A 09 08 |
20 07 06 05 04 03 02 01 00 |
1F
18
17
10
0F
08
07
00

Default Register Bank for R0 – R7

Bank 3

Bank 2

Bank 1

Silicon Labs
<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
<th>PCA0H</th>
<th>PCA0CPH0</th>
<th>PCA0CPH1</th>
<th>PCA0CPH2</th>
<th>PCA0CPH3</th>
<th>PCA0CPH4</th>
<th>WDTCN</th>
<th>BCON1</th>
<th>SBUF1</th>
<th>SADDR1</th>
<th>TL4</th>
<th>TH4</th>
<th>EIP1</th>
<th>EIP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF8</td>
<td>SPI0CN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF0</td>
<td>B</td>
<td>SCON1</td>
<td>SBUF1</td>
<td>SADDR1</td>
<td>TL4</td>
<td>TH4</td>
<td>EIP1</td>
<td>EIP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xE8</td>
<td>ADC0CN</td>
<td>PCA0L</td>
<td>PCA0CPL0</td>
<td>PCA0CPL1</td>
<td>PCA0CPL2</td>
<td>PCA0CPL3</td>
<td>PCA0CPL4</td>
<td>RSTSRC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xE0</td>
<td>ACC</td>
<td>XBR0</td>
<td>XBR1</td>
<td>XBR2</td>
<td>RCAP4L</td>
<td>RCAP4H</td>
<td>EIE1</td>
<td>EIE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xD8</td>
<td>PCA0CN</td>
<td>PCA0MD</td>
<td>PCA0M0</td>
<td>PCA0CPM1</td>
<td>PCA0CPM2</td>
<td>PCA0CPM3</td>
<td>PCA0CPM4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xD0</td>
<td>PSW</td>
<td>REF0CN</td>
<td>DAC0L</td>
<td>DAC0H</td>
<td>DAC0CN</td>
<td>DAC1L</td>
<td>DAC1H</td>
<td>DAC1CN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC8</td>
<td>T2CON</td>
<td>T4CON</td>
<td>RCAP2L</td>
<td>RCAP2H</td>
<td>TL2</td>
<td>TH2</td>
<td>SMB0CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC0</td>
<td>SMB0CN</td>
<td>SMB0ST A</td>
<td>SMB0DAT</td>
<td>SMB0ADR</td>
<td>ADC0GTL</td>
<td>ADC0GTH</td>
<td>ADC0TLH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB8</td>
<td>IP</td>
<td>SADEN0</td>
<td>AMX0CF</td>
<td>AMX0SL</td>
<td>ADC0CF</td>
<td>P1MDIN</td>
<td>ADC0L</td>
<td>ADC0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB0</td>
<td>P3</td>
<td>OSCXCN</td>
<td>OSCICN</td>
<td></td>
<td></td>
<td>P74OUT</td>
<td>FLSC0L</td>
<td>FLACL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA8</td>
<td>IE</td>
<td>SADDR0</td>
<td>ADC1CN</td>
<td>ADC1CF</td>
<td>AMX1SL</td>
<td>P3IF</td>
<td>SADEN1</td>
<td>EMI0CN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA0</td>
<td>P2</td>
<td>EMI0TC</td>
<td></td>
<td>EMI0CF</td>
<td>P0MDOUT</td>
<td>P1MDOUT</td>
<td>P2MDOUT</td>
<td>P3MDOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x98</td>
<td>SCON0</td>
<td>SBUF0</td>
<td>SPI0CFG</td>
<td>SPI0DAT</td>
<td>ADC1</td>
<td>SPI0CKR</td>
<td>CPT0CN</td>
<td>CPT1CN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x90</td>
<td>P1</td>
<td>TMR3CN</td>
<td>TMR3LL</td>
<td>TMR3LH</td>
<td>TMR3L</td>
<td>TMR3H</td>
<td>P7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>TCON</td>
<td>TMOD</td>
<td>TL0</td>
<td>TL1</td>
<td>TH0</td>
<td>TH1</td>
<td>CKCON</td>
<td>PSCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>P0</td>
<td>SP</td>
<td>DPL</td>
<td>DPH</td>
<td>P4</td>
<td>P5</td>
<td>P6</td>
<td>PCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0(8) Bit addressable

1(9) 2(A) 3(B) 4(C) 5(D) 6(E) 7(F)
Addressing Modes

- Eight modes of addressing are available with the C8051F020
- The different addressing modes determine how the operand byte is selected

<table>
<thead>
<tr>
<th>Addressing Modes</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>MOV A, B</td>
</tr>
<tr>
<td>Direct</td>
<td>MOV 30H,A</td>
</tr>
<tr>
<td>Indirect</td>
<td>ADD A,@R0</td>
</tr>
<tr>
<td>Immediate Constant</td>
<td>ADD A,#80H</td>
</tr>
<tr>
<td>Relative*</td>
<td>SJMP AHEAD</td>
</tr>
<tr>
<td>Absolute*</td>
<td>AJMP BACK</td>
</tr>
<tr>
<td>Long*</td>
<td>LJMP FAR_AHEAD</td>
</tr>
<tr>
<td>Indexed</td>
<td>MOVC A,@A+PC</td>
</tr>
</tbody>
</table>

* Related to program branching instructions
The register addressing instruction involves information transfer between registers.

Example:

\[
\text{MOV R0, A}
\]

The instruction transfers the accumulator content into the R0 register. The register bank (Bank 0, 1, 2 or 3) must be specified prior to this instruction.
Direct Addressing

This mode allows you to specify the operand by giving its actual memory address (typically specified in hexadecimal format) or by giving its abbreviated name (e.g. P3)

Note: Abbreviated SFR names are defined in the “C8051F020.inc” header file

Example:

MOV A, P3 ; Transfer the contents of Port 3 to the accumulator

MOV A, 020H ; Transfer the contents of RAM location 20H to the accumulator
Indirect Addressing

- This mode uses a pointer to hold the effective address of the operand
- Only registers R0, R1 and DPTTR can be used as the pointer registers
- The R0 and R1 registers can hold an 8-bit address, whereas DPTTR can hold a 16-bit address

Examples:

```
MOV @R0, A ; Store the content of accumulator into the memory location pointed to by register R0. R0 could have an 8-bit address, such as 60H

MOVX A, @DPTTR ; Transfer the contents from the memory location pointed to by DPTTR into the accumulator. DPTTR could have a 16-bit address, such as 1234H
```
Immediate Constant Addressing

♦ This mode of addressing uses either an 8- or 16-bit constant value as the source operand.
♦ This constant is specified in the instruction, rather than in a register or a memory location.
♦ The destination register should hold the same data size which is specified by the source operand.

♦ Examples:

ADD A, #030H ; Add 8-bit value of 30H to the accumulator register. (which is an 8-bit register).

MOV DPTR, #0FE00H ; Move 16-bit data constant FEOOH into the 16-bit Data Pointer Register.
Relative Addressing

- This mode of addressing is used with some type of jump instructions, like SJMP (short jump) and conditional jumps like JNZ.
- These instructions transfer control from one part of a program to another.
- The destination address must be within -128 and +127 bytes from the current instruction address because an 8-bit offset is used ($2^8 = 256$).

- Example:

  `GoBack: DEC A ;Decrement A
  JNZ GoBack ;If A is not zero, loop back`
Absolute Addressing

♦ Two instructions associated with this mode of addressing are ACALL and AJMP instructions
♦ These are 2-byte instructions where the 11-bit absolute address is specified as the operand
♦ The upper 5 bits of the 16-bit PC address are not modified. The lower 11 bits are loaded from this instruction. So, the branch address must be within the current 2K byte page of program memory ($2^{11} = 2048$)

♦ Example:

```
ACALL PORT_INIT ; PORT_INIT should be located within 2k bytes.
```

```
PORT_INIT: MOV PO, #0FH ; PORT_INIT subroutine
```
Long Addressing

- This mode of addressing is used with the LCALL and LJMP instructions.
- It is a 3-byte instruction and the last 2 bytes specify a 16-bit destination location where the program branches.
- It allows use of the full 64 K code space.
- The program will always branch to the same location no matter where the program was previously.

**Example:**

```
LCALL TIMER_INIT ; TIMER_INIT address (16-bit ; long) is specified as the ; operand; In C, this will be a ; function call: Timer_Init().

TIMER_INIT: ORL TMOD, #01H ; TIMER_INIT subroutine
```
Indexed Addressing

- The Indexed addressing is useful when there is a need to retrieve data from a look-up table.
- A 16-bit register (data pointer) holds the base address and the accumulator holds an 8-bit displacement or index value.
- The sum of these two registers forms the effective address for a JMP or MOVC instruction.

**Example:**

```
MOV A, #08H  ; Offset from table start
MOV DPTR, #01F00H ; Table start address
MOVC A, @A+DPTR  ; Gets target value from the table
                   ; start address + offset and puts it in A
```

- After the execution of the above instructions, the program will branch to address 1F08H (1F00H + 08H) and transfer into the accumulator the data byte retrieved from that location (from the look-up table).