Microcontroller

19. MCS-51 Family uC
Hardware and Software

MCS-51 Family uC
Hardware
8051 Basic Component

- 4K bytes internal ROM
- 128 bytes internal RAM
- Four 8-bit I/O ports (P0 - P3).
- Two 16-bit timers/counters
- One serial interface

8051 Block Diagram

- External Interrupts
- CPU
- OSC
- Bus Control
- 4 I/O Ports
- 4k ROM
- I28 bytes RAM
- Serial
- TXD RXD
- P0 P2 P1 P3
- Addr/Data
- P0 P2 P1 P3

A single chip Microcontroller
Other 8051 Features

- only 1 On chip oscillator (external crystal)
- 6 interrupt sources (2 external, 3 internal, Reset)
- 64K external code (program) memory (only read) PSEN
- 64K external data memory (can be read and write) by RD, WR
- Code memory is selectable by EA (internal or external)
- We may have External memory as data and code

Embedded System

- What is Embedded System?
  - An embedded system is closely integrated with the main system
  - It may not interact directly with the environment
  - For example – A microcomputer in a car ignition control
  - An embedded product uses a microprocessor or microcontroller to do one task only
  - There is only one application software that is typically burned into ROM
Examples of Embedded Systems

- Keyboard
- Printer
- video game player
- MP3 music players
- Embedded memories to keep configuration information
- Mobile phone units
- Domestic (home) appliances
- Data switches
- Automotive controls

Three criteria in Choosing a Microcontroller

- meeting the computing needs of the task efficiently and cost effectively
  - speed, the amount of ROM and RAM, the number of I/O ports and timers, size, packaging, power consumption
  - easy to upgrade
  - cost per unit
- availability of software development tools
  - assemblers, debuggers, C compilers, emulator, simulator, technical support
- wide availability and reliable sources of the microcontrollers
Comparison of the 8051 Family Members

- **ROM type**
  - 8031  no ROM
  - 80xx  mask ROM
  - 87xx  EPROM
  - 89xx  Flash EEPROM

- **89xx**
  - 8951
  - 8952
  - 8953
  - 8955
  - 898252
  - 891051
  - 892051

- **Example (AT89C51, AT89LV51, AT89S51)**
  - AT = ATMEL (Manufacture)
  - C = CMOS technology
  - LV = Low Power (3.0v)

### Comparison of the 8051 Family Members

<table>
<thead>
<tr>
<th>89XX</th>
<th>ROM</th>
<th>RAM</th>
<th>Timer</th>
<th>Int Source</th>
<th>IO pin</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>8951</td>
<td>4k</td>
<td>128</td>
<td>2</td>
<td>6</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>8952</td>
<td>8k</td>
<td>256</td>
<td>3</td>
<td>8</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>8953</td>
<td>12k</td>
<td>256</td>
<td>3</td>
<td>9</td>
<td>32</td>
<td>WD</td>
</tr>
<tr>
<td>8955</td>
<td>20k</td>
<td>256</td>
<td>3</td>
<td>8</td>
<td>32</td>
<td>WD</td>
</tr>
<tr>
<td>898252</td>
<td>8k</td>
<td>256</td>
<td>3</td>
<td>9</td>
<td>32</td>
<td>ISP</td>
</tr>
<tr>
<td>891051</td>
<td>1k</td>
<td>64</td>
<td>1</td>
<td>3</td>
<td>16</td>
<td>AC</td>
</tr>
<tr>
<td>892051</td>
<td>2k</td>
<td>128</td>
<td>2</td>
<td>6</td>
<td>16</td>
<td>AC</td>
</tr>
</tbody>
</table>

WD: Watch Dog Timer  
AC: Analog Comparator  
ISP: In System Programable
8051 DIP40 Pin out

P1.0  P1.1  P1.2  P1.3  P1.4  P1.5  P1.6  P1.7  RST
P1.0  P1.1  P1.2  P1.3  P1.4  P1.5  P1.6  P1.7  RST
1  2  3  4  5  6  7  8
9  10  11  12  13  14  15  16  17
18  19  20  21  22  23  24  25  26
27  28  29  30  31  32  33  34  35
36  37  38  39  40

8051 (8031)  (8751)  (8951)

8051 uC Schematic

[Diagram of 8051 uC Schematic]
**IMPORTANT PINS (I/O Ports)**

- One of the most useful features of the 8051 is that it contains four I/O ports (P0 - P3).

- Port 0: pins 32-39: P0.0 - P0.7
  - 8-bit R/W - General Purpose I/O
  - Or acts as a multiplexed low byte address and data bus for external memory design

- Port 1: pins 1-8: P1.0 - P1.7
  - Only 8-bit R/W - General Purpose I/O

- Port 2: pins 21-28: P2.0 - P2.7
  - 8-bit R/W - General Purpose I/O
  - Or high byte of the address bus for external memory design

- Port 3: pins 10-17: P3.0 - P3.7
  - General Purpose I/O
  - If not using any of the internal peripherals (timers) or external interrupts.

- Each port can be used as input or output (bi-direction)

### Port 3 Alternate Functions

<table>
<thead>
<tr>
<th>Port Pin</th>
<th>Alternate Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3.0</td>
<td>RXD (serial input port)</td>
</tr>
<tr>
<td>P3.1</td>
<td>TXD (serial output port)</td>
</tr>
<tr>
<td>P3.2</td>
<td>INTO (external interrupt 0)</td>
</tr>
<tr>
<td>P3.3</td>
<td>INTO (external interrupt 1)</td>
</tr>
<tr>
<td>P3.4</td>
<td>TO (Timer 0 external input)</td>
</tr>
<tr>
<td>P3.5</td>
<td>T1 (Timer 1 external input)</td>
</tr>
<tr>
<td>P3.6</td>
<td>WR (external data memory write strobe)</td>
</tr>
<tr>
<td>P3.7</td>
<td>RD (external data memory read strobe)</td>
</tr>
</tbody>
</table>
Important Pins

- **PSEN (out):** Program Store Enable, the read signal for external program memory (active low).
- **ALE (out):** Address Latch Enable, to latch address outputs at Port0 and Port2.
- **EA (in):** External Access Enable, active low to access external program memory locations 0 to 4K.
- **RXD, TXD:** UART pins for serial I/O on Port 3.
- **XTAL1 & XTAL2:** Crystal inputs for internal oscillator.

Machine cycle

**Find the machine cycle for:**
- (a) XTAL = 11.0592 MHz
- (b) XTAL = 12 MHz.
- (c) XTAL = 16 MHz.

**Solution:**
- (a) $11.0592 \text{ MHz} / 12 = 921.6 \text{ kHz}$
  - machine cycle $= \frac{1}{921.6 \text{ kHz}} = 1.085 \mu\text{s}$
- (b) $12 \text{ MHz} / 12 = 1 \text{ MHz}$
  - machine cycle $= \frac{1}{1 \text{ MHz}} = 1 \mu\text{s}$
- (b) $16 \text{ MHz} / 12 = 1.333 \text{ MHz}$
  - machine cycle $= \frac{1}{1.333 \text{ MHz}} = 0.75 \mu\text{s}$
RESET Value of Some 8051 Registers:

<table>
<thead>
<tr>
<th>Register</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>0000</td>
</tr>
<tr>
<td>ACC</td>
<td>0000</td>
</tr>
<tr>
<td>B</td>
<td>0000</td>
</tr>
<tr>
<td>PSW</td>
<td>0000</td>
</tr>
<tr>
<td>SP</td>
<td>0007</td>
</tr>
<tr>
<td>DPTR</td>
<td>0000</td>
</tr>
</tbody>
</table>

RAM are all zero
Address Multiplexing for External Memory

(a) Nonmultiplexed (24 pins)

Address Multiplexing for External Memory

(b) Multiplexed (16 pins)

Multiplexing the address (low-byte) and data bus

Accessing external code memory
MCS-51 Memory Structure

On-Chip Memory
Internal RAM

Upper 128 RAM
(Direct and Indirect Addressing)

Special Function
Register's
(Direct Addressing Only)

Lower 128 RAM
(Direct and Indirect Addressing)

Bit Addressable

General Purpose
Registers

Dr. Tuncay UZUN
19. MCS-51 Family uC Hardware and Software
**Registers**

Bank 3
Each bank has R0-R7
Selectable by PSW bit 2,3

Bank 2

Bank 1

Bank 0

**Bit Addressable Memory**

**Byte addressing**: 20h – 2Fh
(16 X 8-bits = 128 bits)

**Bit addressing**:
- `mov C, 1Ah`
- or
- `mov C, 23h.2`
Special Function Registers

- DATA registers
- CONTROL registers
  - Timers
  - Serial ports
  - Interrupt system
  - Analog to Digital converter
  - Digital to Analog converter
  - Etc.

Addresses 80h – FFh
Direct Addressing used to access SFRs

Bit Addressable RAM

Summary of the 8051 on-chip data memory (RAM)
Bit Addressable RAM (cont.)

Summary of the 8051 on-chip data memory (SFR)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>Bit address</th>
<th>Byte address</th>
<th>Bit address</th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td>FF</td>
<td>F0</td>
<td>FF</td>
</tr>
<tr>
<td>90</td>
<td>FF</td>
<td>F0</td>
<td>FF</td>
</tr>
<tr>
<td>8D</td>
<td>not bit addressable</td>
<td>TH1</td>
<td>ACC</td>
</tr>
<tr>
<td>8C</td>
<td>not bit addressable</td>
<td>TH0</td>
<td>D0</td>
</tr>
<tr>
<td>8B</td>
<td>not bit addressable</td>
<td>TL1</td>
<td>D1</td>
</tr>
<tr>
<td>8A</td>
<td>not bit addressable</td>
<td>TL0</td>
<td>B8</td>
</tr>
<tr>
<td>89</td>
<td>not bit addressable</td>
<td>TMOD</td>
<td>B9</td>
</tr>
<tr>
<td>88</td>
<td>not bit addressable</td>
<td>TCON</td>
<td>B0</td>
</tr>
<tr>
<td>87</td>
<td>not bit addressable</td>
<td>PCON</td>
<td>A8</td>
</tr>
<tr>
<td>83</td>
<td>not bit addressable</td>
<td>DPH</td>
<td>A9</td>
</tr>
<tr>
<td>82</td>
<td>not bit addressable</td>
<td>DPL</td>
<td>A0</td>
</tr>
<tr>
<td>81</td>
<td>not bit addressable</td>
<td>SP</td>
<td>A7</td>
</tr>
<tr>
<td>80</td>
<td>not bit addressable</td>
<td>P0</td>
<td>99</td>
</tr>
</tbody>
</table>

SFR MEMORY MAP

<table>
<thead>
<tr>
<th>F6</th>
<th>F0</th>
<th>E8</th>
<th>E0</th>
<th>D8</th>
<th>D0</th>
<th>C6</th>
<th>C0</th>
<th>B6</th>
<th>B0</th>
<th>A6</th>
<th>A0</th>
<th>96</th>
<th>90</th>
<th>88</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td>ACC</td>
<td>PSW</td>
<td></td>
<td>T2CON</td>
<td>RCAP2L</td>
<td>RCAP2H</td>
<td>TL2</td>
<td>TH2</td>
<td></td>
<td></td>
<td>SCON</td>
<td>SBUF</td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
</tr>
<tr>
<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></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5
MCS-51 Family uC Software

CPU Registers

- **A**: (8-bit Accumulator)
- **B**: (8-bit B register)
- **PSW**: (8-bit Program Status Word)
- **SP**: (16-bit Stack Pointer)
- **PC**: (16-bit Program Counter)
- **DPTR**: (16-bit Data Pointer)

Used in assembler instructions
R0-R7 RAM Registers

Four Register Banks
Each bank has R0-R7
Selectable by PSW bit 2,3

Bank 3
Bank 2
Bank 1
Bank 0

PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

<table>
<thead>
<tr>
<th>CY</th>
<th>AC</th>
<th>F0</th>
<th>RS1</th>
<th>RS0</th>
<th>OV</th>
<th>—</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>CY</td>
<td>PSW.7 Carry Flag.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>PSW.6 Auxiliary Carry Flag.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0</td>
<td>PSW.5 Flag 0 available to the user for general purpose.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS1</td>
<td>PSW.4 Register Bank selector bit 1 (SEE NOTE 1).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS0</td>
<td>PSW.3 Register Bank selector bit 0 (SEE NOTE 1).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OV</td>
<td>PSW.2 Overflow Flag.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>—</td>
<td>PSW.1 User definable flag.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>PSW.0 Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE:
1. The value presented by RS0 and RS1 selects the corresponding register bank.

<table>
<thead>
<tr>
<th>RS1</th>
<th>RS0</th>
<th>Register Bank</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00H-07H</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>08H-0FH</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>10H-17H</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>18H-1FH</td>
</tr>
</tbody>
</table>
MCS-51 Assembly Language

MCS-51 Instruction Structure
MOV destination, source ; dest. ← source

<table>
<thead>
<tr>
<th>6802 uP</th>
<th>8051 uC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAA #40H</td>
<td>MOV A,#40H</td>
</tr>
<tr>
<td>LDAA 40H</td>
<td>MOV A,40H</td>
</tr>
<tr>
<td>STAA 41H</td>
<td>MOV 40H,A</td>
</tr>
<tr>
<td>LDAA 0,X</td>
<td>MOVC A,@A+DPTR</td>
</tr>
<tr>
<td>BCS L1</td>
<td>JC L1</td>
</tr>
<tr>
<td>JSR 1200H</td>
<td>LCALL 1200H</td>
</tr>
<tr>
<td>CLC</td>
<td>CLR C</td>
</tr>
</tbody>
</table>

Addressing Modes

**Immediate Mode** – specify data by its value

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV A, #0</td>
<td>put 0 in the accumulator</td>
<td>A = 00000000</td>
</tr>
<tr>
<td>MOV R4, #11h</td>
<td>put 11hex in the R4 register</td>
<td>R4 = 00010001</td>
</tr>
<tr>
<td>MOV B, #11</td>
<td>put 11 decimal in b register</td>
<td>B = 00001011</td>
</tr>
<tr>
<td>MOV DPTR, #7521h</td>
<td>put 7521 hex in DPTR</td>
<td>DPTR = 0111010100100001</td>
</tr>
</tbody>
</table>
Addressing Modes

Register Addressing – either source or destination is one of CPU register

MOV R0, A
MOV A, R7
ADD A, R4
ADD A, R7
MOV DPTR, #25F5H
MOV R5, DPL
MOV R, DPH

Note: that MOV R4, R7 is incorrect

Direct Mode – specify data by its 8-bit address usually for 30h-7Fh of RAM

MOV A, 70h ; copy contents of RAM at 70h to a
MOV R0, 40h ; copy contents of RAM at 70h to a
MOV 56h, A ; put contents of a at 56h to a
MOV 0D0h, A ; put contents of a into PSW
Addressing Modes

Register Indirect – the address of the source or destination is specified in registers

Uses registers R0 or R1 for 8-bit address:

```
MOV PSW,#0 ; use register bank 0
MOV R0,#0x3C
MOV @R0,#3 ; memory at 3C gets #3
    ; M[3C] ← 3
```

Uses DPTR register for 16-bit addresses:

```
MOV DPTR,#0x9000 ; DPTR ← 9000h
MOVX A,@DPTR ; A ← M[9000]
```

Note that 9000 is an address in external memory

Use Register Indirect to access upper RAM block (+8052)
Addressing Modes

**Register Indexed Mode** – source or destination address is the sum of the base address and the accumulator (Index)
- Base address can be DPTR or PC
  - MOV DPTR,#4000h
  - MOV A,#5
  - MOV A,@A+DPTR ; a ← M[4005]

**Addressing Modes**

**Register Indexed Mode** continue

Base address can be DPTR or PC

```
ORG 1000h
1000 MOV A,#5
1002 MOVCA @A+PC ; a ← M[1008]
1003 NOP
```

- Table Lookup
- MOVC only can read internal code memory
**MCS-51 Instruction Set**

- Arithmetic Operation Instructions
- Logic Operation Instructions
- Data Transfer Instructions
- Boolean Variable Manipulation Instructions
- Program Branching Instructions

**Data Transfer Instructions**

- MOV dest, source ; dest ← source
- Stack instructions
  - PUSH byte ; increment stack pointer,
    ; move byte on stack
  - POP byte ; move from stack to byte,
   ; decrement stack pointer
- Exchange instructions
  - XCH a,byte ; exchange accumulator and byte
  - XCHD a,byte ; exchange low nibbles of
    ; accumulator and byte
### Acc Register

- A register can be accessed by **direct** and **register** mode.
- This 3 instructions have **same** function with **different** code.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0703 E500</td>
<td>mov a, 00h</td>
</tr>
<tr>
<td>0705 8500E0</td>
<td>mov acc, 00h</td>
</tr>
<tr>
<td>0708 8500E0</td>
<td>mov 0e0h, 00h</td>
</tr>
</tbody>
</table>

- Also this 3 instruction

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>070B E9</td>
<td>mov a, r1</td>
</tr>
<tr>
<td>070C 89E0</td>
<td>mov acc, r1</td>
</tr>
<tr>
<td>070E 89E0</td>
<td>mov 0e0h, r1</td>
</tr>
</tbody>
</table>

### SFRs Address

- **B** – always direct mode - except in MUL & DIV

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0703 8500F0</td>
<td>mov b, 00h</td>
</tr>
<tr>
<td>0706 8500F0</td>
<td>mov 0f0h, 00h</td>
</tr>
<tr>
<td>0709 8CF0</td>
<td>mov b, r4</td>
</tr>
<tr>
<td>070B 8CF0</td>
<td>mov 0f0h, r4</td>
</tr>
</tbody>
</table>

- **P0~P3** – are direct address

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0704 F580</td>
<td>mov p0, a</td>
</tr>
<tr>
<td>0706 F580</td>
<td>mov 80h, a</td>
</tr>
<tr>
<td>0708 859080</td>
<td>mov p0, p1</td>
</tr>
</tbody>
</table>

- Also other SFRs (pcon, tmod, psw,...)
8051 Instruction Format

- immediate addressing

<table>
<thead>
<tr>
<th>Op code</th>
<th>Immediate data</th>
</tr>
</thead>
<tbody>
<tr>
<td>add a,#3dh</td>
<td>;machine code=243d</td>
</tr>
</tbody>
</table>

- Direct addressing

<table>
<thead>
<tr>
<th>Op code</th>
<th>Direct address</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov r3,0E8h</td>
<td>;machine code=ABE8</td>
</tr>
</tbody>
</table>

8051 Instruction Format

- Register addressing

<table>
<thead>
<tr>
<th>Op code</th>
<th>n n n</th>
</tr>
</thead>
<tbody>
<tr>
<td>070D E8</td>
<td>mov a,r0 ;E8 = 1110 1000</td>
</tr>
<tr>
<td>070E E9</td>
<td>mov a,r1 ;E9 = 1110 1001</td>
</tr>
<tr>
<td>070F EA</td>
<td>mov a,r2 ;EA = 1110 1010</td>
</tr>
<tr>
<td>0710 ED</td>
<td>mov a,r5 ;ED = 1110 1101</td>
</tr>
<tr>
<td>0711 EF</td>
<td>mov a,r7 ;Ef = 1110 1111</td>
</tr>
<tr>
<td>0712 2F</td>
<td>add a,r7</td>
</tr>
<tr>
<td>0713 F8</td>
<td>mov r0,a</td>
</tr>
<tr>
<td>0714 F9</td>
<td>mov r1,a</td>
</tr>
<tr>
<td>0715 FA</td>
<td>mov r2,a</td>
</tr>
<tr>
<td>0716 FD</td>
<td>mov r5,a</td>
</tr>
<tr>
<td>0717 FD</td>
<td>mov r5,a</td>
</tr>
</tbody>
</table>
8051 Instruction Format

• Register indirect addressing

<table>
<thead>
<tr>
<th>Op code</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov a, @Ri</td>
<td>; i = 0 or 1</td>
</tr>
<tr>
<td>070D E7</td>
<td>mov a, @r1</td>
</tr>
<tr>
<td>070D 93</td>
<td>movc a, @a+dptr</td>
</tr>
<tr>
<td>070E 83</td>
<td>movc a, @a+pc</td>
</tr>
<tr>
<td>070F E0</td>
<td>movx a, @dptr</td>
</tr>
<tr>
<td>0710 F0</td>
<td>movx @dptr, a</td>
</tr>
<tr>
<td>0711 F2</td>
<td>movx @r0, a</td>
</tr>
<tr>
<td>0712 E3</td>
<td>movx a, @r1</td>
</tr>
</tbody>
</table>

8051 Instruction Format

• Relative addressing

<table>
<thead>
<tr>
<th>Op code</th>
<th>Relative address</th>
</tr>
</thead>
<tbody>
<tr>
<td>here: SJMP here</td>
<td>machine code=80FE (FE=-2)</td>
</tr>
<tr>
<td>Range = (-128 ~ 127)</td>
<td></td>
</tr>
</tbody>
</table>

• Absolute addressing (limited in 2k current mem block)

<table>
<thead>
<tr>
<th>A10-A8</th>
<th>Op code</th>
<th>A7-A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0700</td>
<td>1</td>
<td>org 0700h</td>
</tr>
<tr>
<td>0700 E106</td>
<td>2</td>
<td>ajmp next ; next=706h</td>
</tr>
<tr>
<td>0702 00</td>
<td>3</td>
<td>nop</td>
</tr>
<tr>
<td>0703 00</td>
<td>4</td>
<td>nop</td>
</tr>
<tr>
<td>0704 00</td>
<td>5</td>
<td>nop</td>
</tr>
<tr>
<td>0705 00</td>
<td>6</td>
<td>next:</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>end</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8051 Instruction Format

- Long distance address

<table>
<thead>
<tr>
<th>Op code</th>
<th>A15-A8</th>
<th>A7-A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range = (0000h ~ FFFFh)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0700 1 org 0700h
0700 020707 2 ajmp next
0703 00 3 nop
0704 00 4 nop
0705 00 5 nop
0706 00 6 nop
7 next:
8 end

Stack & Stack Pointer (SP)
### Stack

- Stack-oriented data transfer
  - Only one operand (direct addressing)
  - SP is other operand – register indirect - implied
- Direct addressing mode must be used in PUSH and POP

```asm
MOV SP,#0x40 ; Initialize SP
PUSH 0x55 ; SP ← SP+1, M[SP] ← M[55]
           ; M[41] ← M[55]
POP B     ; B ← M[55]
```

Note: can only specify RAM or SFRs (direct mode) to push or pop. Therefore, to push/pop the accumulator, must use acc, not a

### Multiply

When multiplying two 8-bit numbers, the size of the maximum product is 16-bits

\[
\text{FF} \times \text{FF} = \text{FE01} \\
(255 \times 255 = 65025)
\]

```asm
MUL AB    ; BA ← A \times B
```

Note: B gets the High byte
A gets the Low byte
Division

• Integer Division

\[ \text{DIV AB } ; \text{ divide A by B} \]

\[ \text{A } \leftarrow \text{Quotient(A/B)} \]
\[ \text{B } \leftarrow \text{Remainder(A/B)} \]

OV - used to indicate a divide by zero condition.
C – set to zero

Decimal Adjust Accum. for Add.

\[ \text{DA A } ; \text{ decimal adjust a} \]

Used to facilitate BCD addition.
Adds "6" to either high or low nibble after an addition to create a valid BCD number.

Example:
\[
\begin{align*}
\text{mov a,}\#23h \\
\text{mov b,}\#29h \\
\text{add a,b} ; \text{ a } \leftarrow 23h + 29h = 4Ch \text{ (wanted 52)} \\
\text{DA a} ; \text{ a } \leftarrow \text{ a + 6 = 52}
\end{align*}
\]
Rotate

- Rotate instructions operate **only on a**

\[
\text{RL } a \\
\text{Mov } a, \#0xF0 \quad ; \quad a \leftarrow 11110000 \\
\text{RR } a \quad ; \quad a \leftarrow 11100001
\]

\[
\text{RR } a \\
\text{Mov } a, \#0xF0 \quad ; \quad a \leftarrow 11110000 \\
\text{RR } a \quad ; \quad a \leftarrow 01111000
\]

Rotate through Carry

\[
\text{RRC } a \\
\text{mov } a, \#0A9h \quad ; \quad a \leftarrow A9 \\
\text{add } a, \#14h \quad ; \quad a \leftarrow BD \ (10111101), \ C \leftarrow 0 \\
\text{rrc } a \quad ; \quad a \leftarrow 01011110, \ C \leftarrow 1
\]

\[
\text{RLC } a \\
\text{mov } a, \#3ch \quad ; \quad a \leftarrow 3ch(00111100) \\
\text{setb } c \quad ; \quad c \leftarrow 1 \\
\text{rlc } a \quad ; \quad a \leftarrow 01110001, \ C \leftarrow 1
\]
Swap

\[
\text{SWAP a}
\]

\[
\begin{align*}
\text{mov a,} & \#72h \\
\text{swap a} & \\
\end{align*}
\]

; \ a & \leftrightarrow \ 27h

Conditional Jump

- These instructions cause a jump to occur \textbf{only} if a condition is true. Otherwise, program execution continues with the next instruction.

\[
\text{loop: mov a, P1}
\]

\[
\begin{align*}
\text{jz loop} & \quad ; \text{if a=0, goto loop,} \\
\text{mov b, a} & \quad ; \text{else goto next instruction}
\end{align*}
\]

- There is \textbf{no} zero flag (\textbf{z})
- Content of A checked for zero \textbf{on time}
### Conditional jumps

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JZ &lt;rel addr&gt;</td>
<td>Jump if a = 0</td>
</tr>
<tr>
<td>JNZ &lt;rel addr&gt;</td>
<td>Jump if a != 0</td>
</tr>
<tr>
<td>JC &lt;rel addr&gt;</td>
<td>Jump if C = 1</td>
</tr>
<tr>
<td>JNC &lt;rel addr&gt;</td>
<td>Jump if C != 1</td>
</tr>
<tr>
<td>JB &lt;bit&gt;,&lt;rel addr&gt;</td>
<td>Jump if bit = 1</td>
</tr>
<tr>
<td>JNB &lt;bit&gt;,&lt;rel addr&gt;</td>
<td>Jump if bit != 1</td>
</tr>
<tr>
<td>JBC &lt;bit&gt;,&lt;rel addr&gt;</td>
<td>Jump if bit =1, &amp;clear bit</td>
</tr>
<tr>
<td>CJNE A,direct,&lt;rel addr&gt;</td>
<td>Compare A and memory, jump if not equal</td>
</tr>
</tbody>
</table>

#### Example: Conditional Jumps

```assembly
if (a = 0) is true
    send a 0 to LED
else
    send a 1 to LED

jz led_off
Setb P1.6
sjmp skipover
led_off: clr P1.6
mov A, P0
skipover:
```

if (a = 0) is true
    send a 0 to LED
else
    send a 1 to LED

jz led_off
Setb P1.6
sjmp skipover
led_off: clr P1.6
mov A, P0
skipover:
Iterative Loops

For A = 0 to 4 do
{...
    clr a
    loop: ...
    ...
    inc a
    cjne a, #4,
    loop
}

For A = 4 to 0 do
{...
    mov R0, #4
    loop: ...
    ...
    djnz R0, loop
}

Call and Return

• Call is similar to a jump, but
  – Call pushes PC on stack before branching

    acall <address 11> ; stack ← PC
                       ; PC ← address 11 bit

    lcall <address 16> ; stack ← PC
                       ; PC ← address 16 bit

• Return is also similar to a jump, but
  – Return instruction pops PC from stack to get address to
    jump to

    ret ; PC ← stack
Interrupt Vectors

Each interrupt has a **specific** place in **code** memory where program execution (interrupt service routine) begins.

- External Interrupt 0 : 0003h
- Timer 0 overflow : 000Bh
- External Interrupt 1 : 0013h
- Timer 1 overflow : 001Bh
- Serial : 0023h
- Timer 2 overflow (8052+) : 002Bh

**Note:** that there are only 8 memory locations between vectors.

Interrupt Vectors

To avoid overlapping Interrupt Service routines, it is common to put **JUMP** instructions at the vector address. This is similar to the reset vector.

```
org 009BH ; at EX7 vector
ljmp EX7ISR
```

```
cseg at 0x100; at Main program
Main: ... ; Main program
...
EX7ISR:... ; Interrupt service routine
... ; Can go after main program
reti ; and subroutines.
```