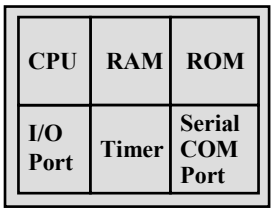
Microcontroller

19. MCS-51 Family uC Hardware and Software

MCS-51 Family uC Hardware

8051 Basic Component

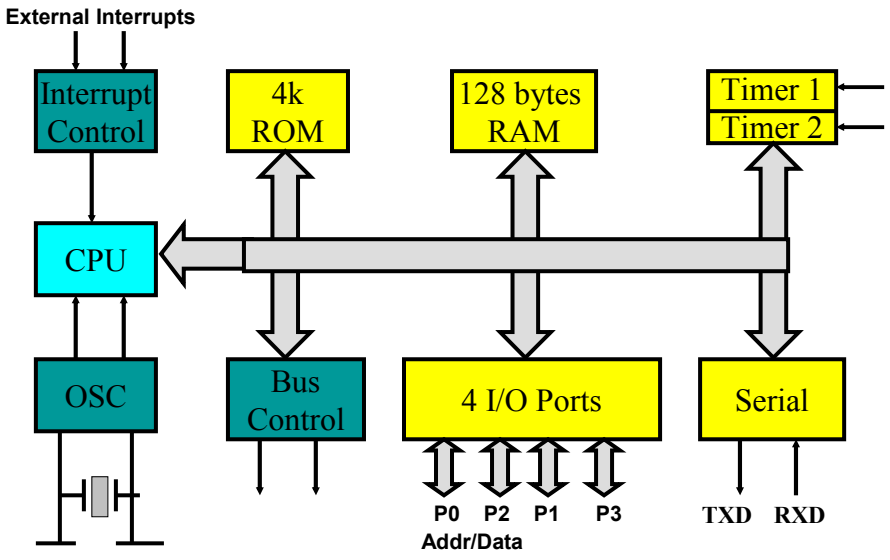
- 4K bytes internal ROM
- 128 bytes internal RAM
- Four 8-bit I/O ports (P0 - P3).
- Two 16-bit timers/counters
- One serial interface



A single chip Microcontroller

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 3

8051 Block Diagram



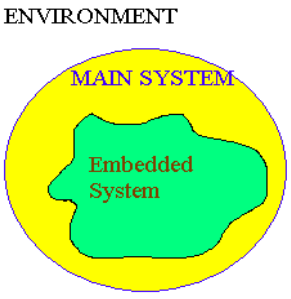
Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 4

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)

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 9

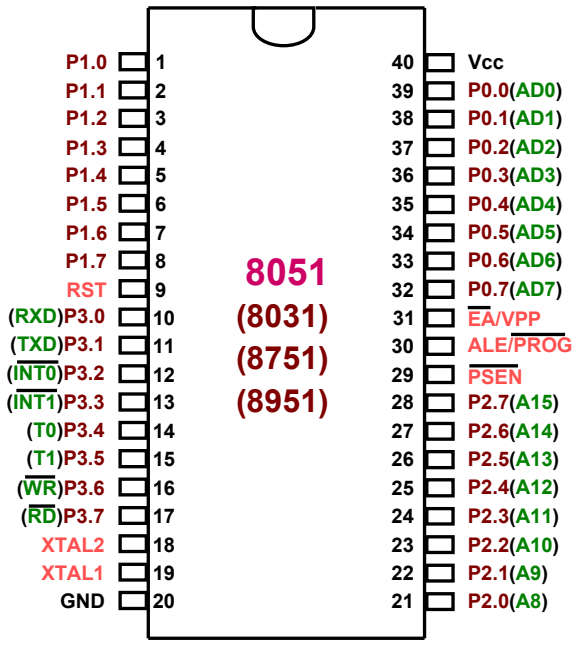
Comparison of the 8051 Family Members

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

WD: Watch Dog Timer
AC: Analog Comparator
ISP: In System Programable

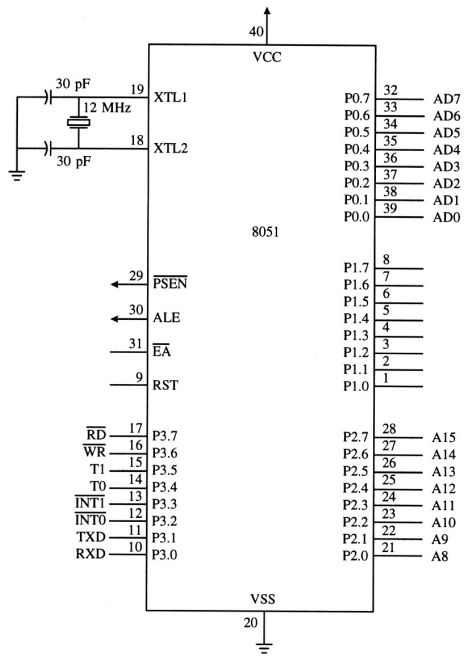
Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 10

8051 DIP40 Pin out



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 11

8051 uC Schematic



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 12

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 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 P1.0 P1.7
  - Only 8-bit R/W - General Purpose I/O
- Port 2 pins 21-28 P2 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 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)**

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 13

Port 3 Alternate Functions

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 14

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.

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 15

Machine cycle

Find the machine cycle for:

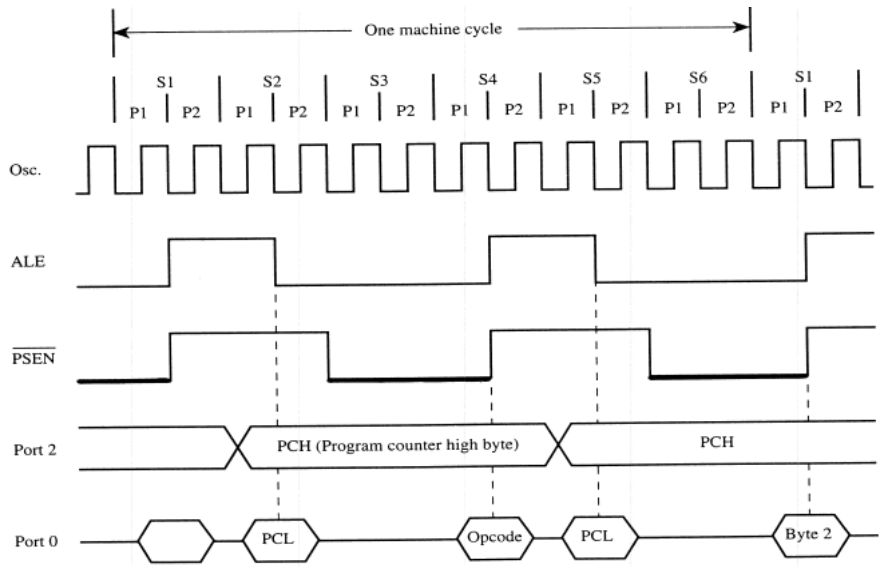
- (a) XTAL = 11.0592 MHz
- (b) XTAL = 12 MHz.
- (c) XTAL = 16 MHz.

Solution:

- (a) 11.0592 MHz / 12 = 921.6 kHz;
- machine cycle = 1 / 921.6 kHz = 1.085 μs
- (b) 12 MHz / 12 = 1 MHz;
- machine cycle = 1 / 1 MHz = 1 μs
- (b) 16 MHz / 12 = 1.333 MHz;
- machine cycle = 1 / 1.333 MHz = 0.75 μs

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 16

Read timing for external code memory



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 17

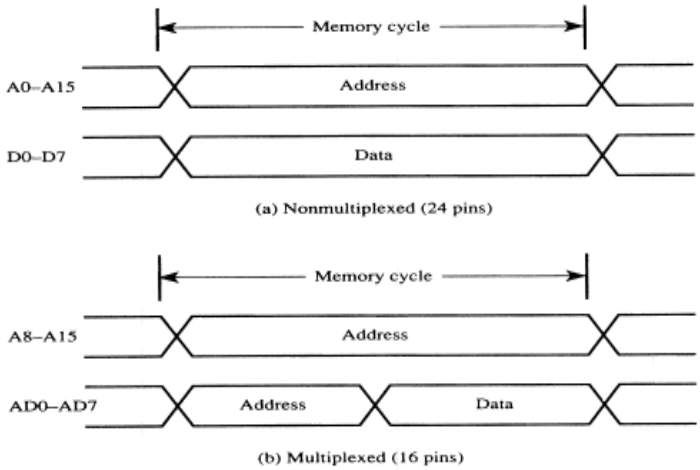
RESET Value of Some 8051 Registers:

Register	Reset Value
PC	0000
ACC	0000
B	0000
PSW	0000
SP	0007
DPTR	0000

RAM are all zero

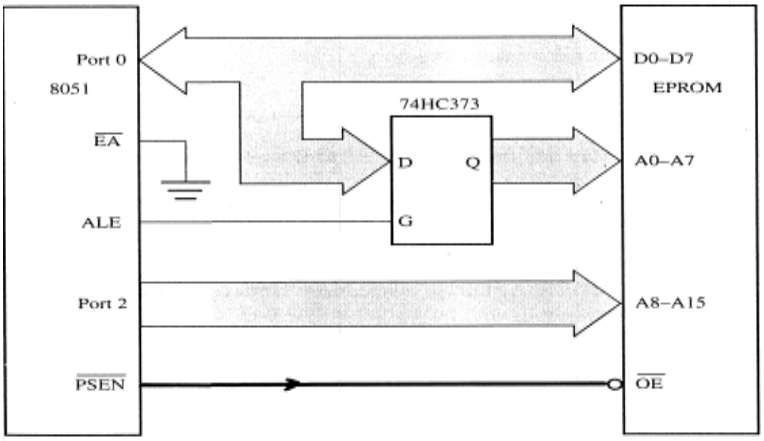
Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 18

Address Multiplexing for External Memory



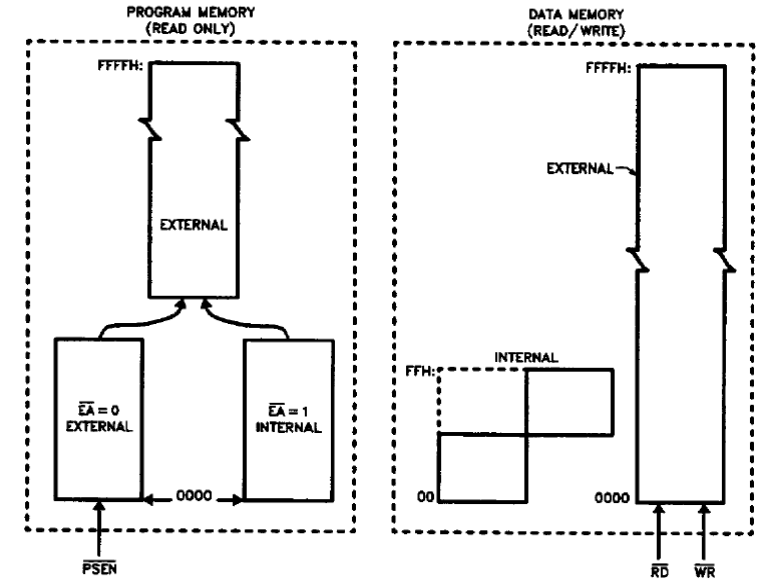
Multiplexing the address (low-byte) and data bus

Address Multiplexing for External Memory



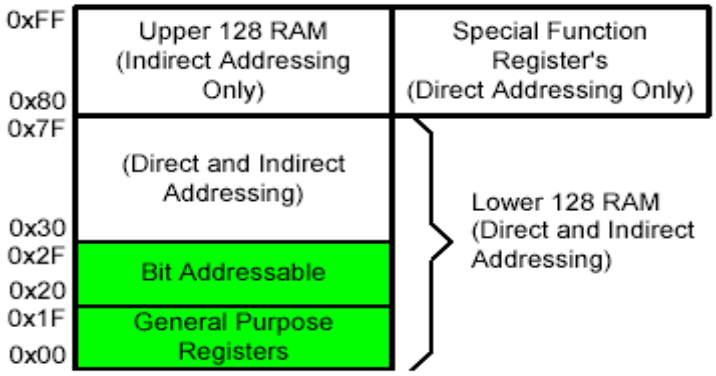
Accessing external code memory

MCS-51 Memory Structure



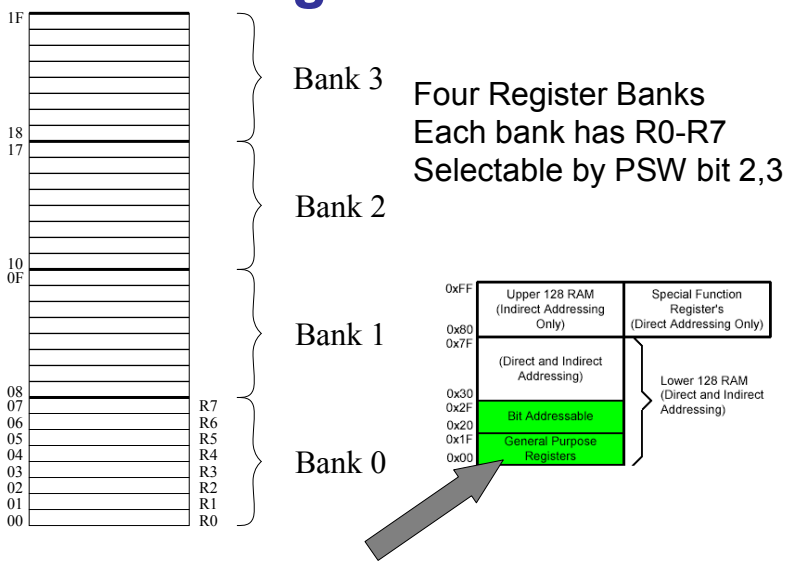
Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 21

On-Chip Memory
Internal RAM



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 22

Registers



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 23

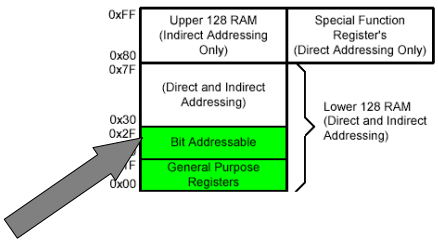
Bit Addressable Memory

2F	7F							78
2E								
2D								
2C								
2B								
2A								
29								
28								
27								
26								
25								
24								
23						1A		
22								10
21	0F							08
20	07	06	05	04	03	02	01	00

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

Bit addressing:

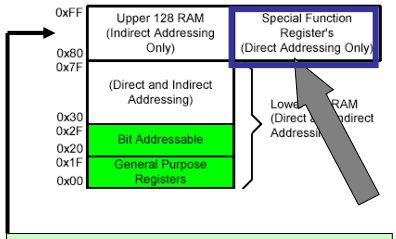
mov C, **1A**h
or
mov C, **23**h.2



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 24

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

RAM

Byte address	Bit address							
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

Bit-addressable locations

1F 18	Bank 3
17 10	Bank 2
0F 08	Bank 1
07 00	Default register bank for R0–R7

Byte address	Bit address							
7F	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

Bit-addressable locations

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

Bit Addressable RAM (cont.)

Byte address	Bit address									Byte address	Bit address								
98	9F	9E	9D	9C	9B	9A	99	98	SCON	FF									
										F0	F7	F6	F5	F4	F3	F2	F1	F0	B
90	97	96	95	94	93	92	91	90	P1										
										E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
8D	not bit addressable								TH1										
8C	not bit addressable								TH0	D0	D7	D6	D5	D4	D3	D2	–	D0	PSW
8B	not bit addressable								TL1										
8A	not bit addressable								TL0	B8	–	–	–	BC	BB	BA	B9	B8	IP
89	not bit addressable								TMOD										
88	8F	8E	8D	8C	8B	8A	89	88	TCON	B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
87	not bit addressable								PCON										
										A8	AF	–	–	AC	AB	AA	A9	A8	IE
83	not bit addressable								DPH										
82	not bit addressable								DPL	A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
81	not bit addressable								SP										
80	87	86	85	84	83	82	81	80	P0	99	not bit addressable								SBUF

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

SFR MEMORY MAP

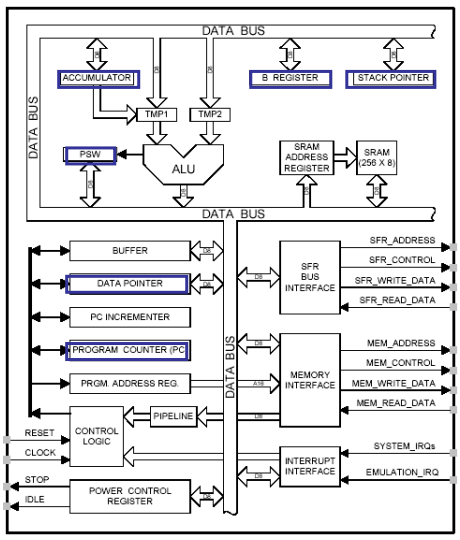
	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

↑ Bit Addressable

Figure 5

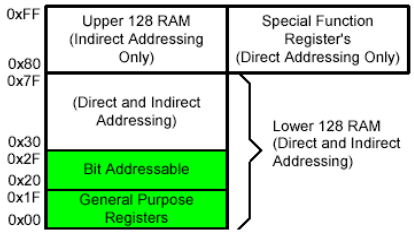
MCS-51 Family uC Software

MCS-51 Programming Model



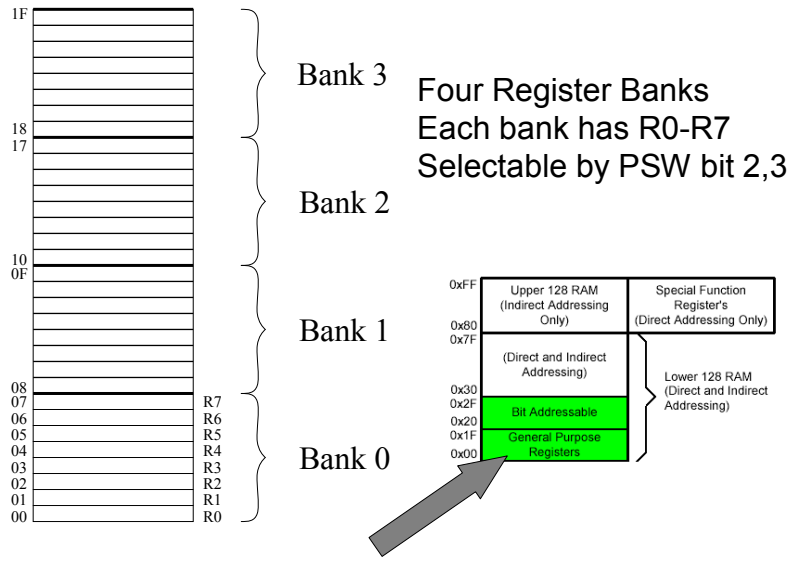
CPU Registers

- 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



PSW: PROGRAM STATUS WORD. BIT ADDRESSABLE.

CY	AC	F0	RS1	RS0	OV	—	P

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

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

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

MCS-51 Assembly Language

MCS-51 Instruction Structure

MOV destination,source ; dest. ← source

6802 uP		8051 uC	
LDAA	#40H	MOV	A,#40H
LDAA	40H	MOV	A,40H
STAA	41H	MOV	40H,A
LDAA	0,X	MOVC	A,@A+DPTR
BCS	L1	JC	L1
JSR	1200H	LCALL	1200H
CLC		CLR	C

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 33

Addressing Modes

Immediate Mode – specify data by its value

```
MOV A,#0        ;put 0 in the accumulator
                ;A = 00000000

MOV R4,#11h     ;put 11hex in the R4 register
                ;R4 = 00010001

MOV B,#11       ;put 11 decimal in b register
                ;B = 00001011

MOV DPTR,#7521h ;put 7521 hex in DPTR
                ;DPTR = 0111010100100001
```

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 34

Addressing Modes

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

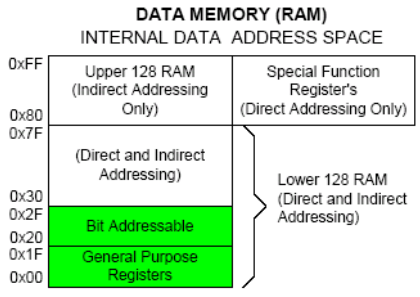
```
MOV R0,A
MOV A,R7
ADD A,R4
ADD A,R7
MOV DPTR,#25F5H
MOV R5,DPL
MOV R,DPH
```

Note: that MOV R4,R7 is incorrect

Addressing Modes

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
```

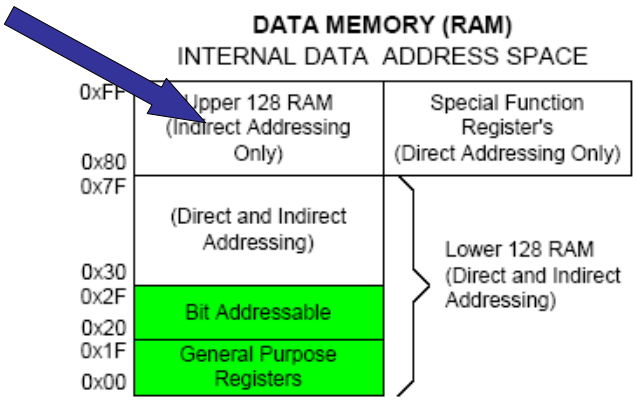
Uses DPTR register for 16-bit addresses:

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

Note that 9000 is an address in external memory

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 37

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



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 38

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
PC → 1002 MOVC A,@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 instruction has same function with different code

```
0703 E500        mov a,00h
0705 8500E0      mov acc,00h
0708 8500E0      mov 0e0h,00h
```

- Also this 3 instruction

```
070B E9          mov a,r1
070C 89E0        mov acc,r1
070E 89E0        mov 0e0h,r1
```

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 43

SFRs Address

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

```
0703 8500F0      mov b,00h
0706 8500F0      mov 0f0h,00h

0709 8CF0        mov b,r4
070B 8CF0        mov 0f0h,r4
```

- P0~P3 – are direct address

```
0704 F580        mov p0,a
0706 F580        mov 80h,a
0708 859080      mov p0,p1
```

- Also other SFRs (pcon, tmod, psw,….)

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 44

8051 Instruction Format

- immediate addressing

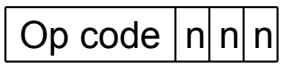

```
add a,#3dh        ;machine code=243d
```

- Direct addressing


```
mov r3,0E8h       ;machine code=ABE8
```


8051 Instruction Format

- Register addressing




```
070D E8     mov a,r0        ;E8 = 1110 1000
070E E9     mov a,r1        ;E9 = 1110 1001
070F EA     mov a,r2        ;EA = 1110 1010
0710 ED     mov a,r5        ;ED = 1110 1101
0711 EF     mov a,r7        ;Ef = 1110 1111
0712 2F     add a,r7
0713 F8     mov r0,a
0714 F9     mov r1,a
0715 FA     mov r2,a
0716 FD     mov r5,a
0717 FD     mov r5,a
```

8051 Instruction Format

- Register indirect addressing

Op code	i

mov a, @Ri ; i = 0 or 1

```
070D E7      mov  a,@r1
070D 93      movc a,@a+dptr
070E 83      movc a,@a+pc
070F E0      movx a,@dptr
0710 F0      movx @dptr,a
0711 F2      movx @r0,a
0712 E3      movx a,@r1
```


8051 Instruction Format

- relative addressing

Op code	Relative address

here: SJMP here ;machine code=80FE (FE=-2)

Range = (-128 ~ 127)

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

A10-A8	Op code	A7-A0

```
0700                1       org 0700h
0700 E106           2       ajmp next    ;next=706h
0702 00             3       nop
0703 00             4       nop
0704 00             5       nop
0705 00             6       nop
                    7   next:
                    8       end
```

07FEh

8051 Instruction Format

- Long distance address

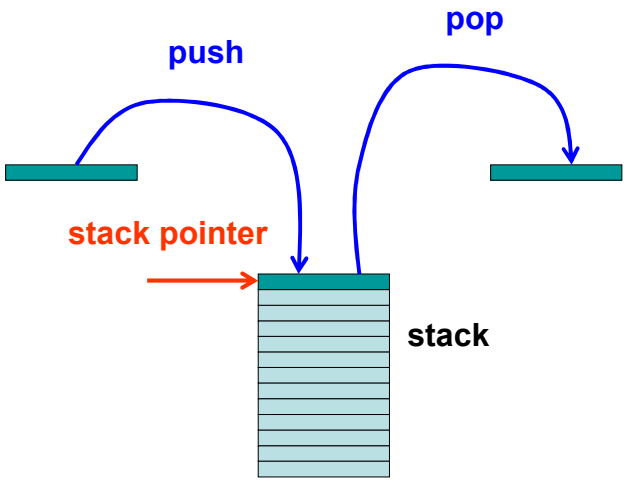
Op code	A15-A8	A7-A0

Range = (0000h ~ FFFFh)

```
0700                    1           org 0700h
0700 020707             2           ajmp next
  ;next=0707h
0703 00                 3           nop
0704 00                 4           nop
0705 00                 5           nop
0706 00                 6           nop
                        7     next:
                        8           end
```

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 49

Stack & Stack Pointer (SP)



Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 50

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

```
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

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

Multiply

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

FF x FF = FE01

(255 x 255 = 65025)

```
MUL AB      ; BA ← A * B
```

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

Dr.Tuncay UZUN 19. MCS-51 Family uC Hardware and Software 52

Division

- Integer Division

```
DIV AB   ; divide A by B

A ← Quotient(A/B)
B ← Remainder(A/B)
```

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

Decimal Adjust Accum. for Add.

```
DA A        ; 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.

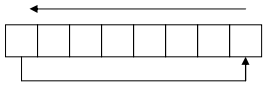
Example:

```
mov a,#23h
mov b,#29h
add a,b     ; a ← 23h + 29h = 4Ch (wanted 52)
DA  a       ; a ← a + 6 = 52
```

Rotate

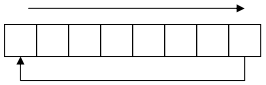
- Rotate instructions operate only on **a**

RL a



```
Mov a,#0xF0   ; a← 11110000
RR a          ; a← 11100001
```

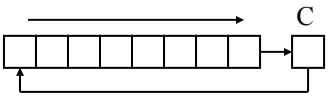
RR a



```
Mov a,#0xF0   ; a← 11110000
RR a          ; a← 01111000
```

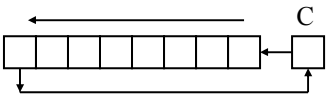

Rotate through Carry

RRC a




```
mov a, #0A9h   ; a ← A9
add a, #14h    ; a ← BD (10111101), C←0
rrc a          ; a ← 01011110, C←1
```

RLC a




```
mov a, #3ch    ; a ← 3ch(00111100)
setb c         ; c ← 1
rlc a          ; a ← 01111001, C←1
```

Swap

SWAP a

```
mov  a,#72h     ; a ← 27h
swap a          ; a ← 27h
```


Conditional Jump

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

```
loop: mov a, P1
      jz  loop     ; if a=0, goto loop,
                   ; else goto next instruction
      mov b, a
```

- There is no zero flag (z)
- Content of A checked for zero on time

Conditional jumps

Mnemonic	Description
JZ <rel addr>	Jump if a = 0
JNZ <rel addr>	Jump if a != 0
JC <rel addr>	Jump if C = 1
JNC <rel addr>	Jump if C != 1
JB <bit>,<rel addr>	Jump if bit = 1
JNB <bit>,<rel addr>	Jump if bit != 1
JBC <bir>,<rel addr>	Jump if bit =1, &clear bit
CJNE A,direct,<rel addr>	Compare A and memory, jump if not equal

Example: Conditional Jumps

```
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.
```