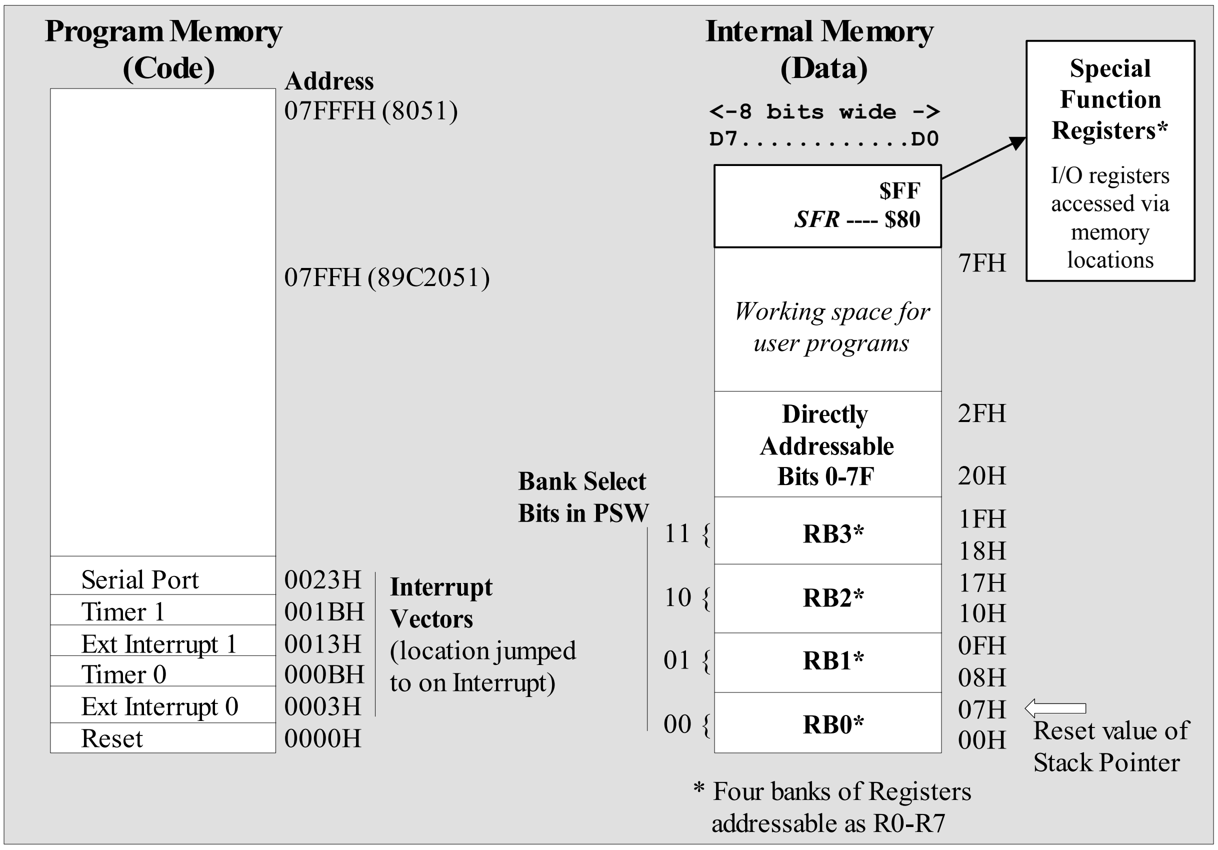
8051 Architecture Reference v5



Program Counter:

```
16 bit register restricted to 0000H -> 07FFFH
```

*Special Function Registers (SFR) Space:

Byte Address	Name	Description	Bits ("x" => NOT bit addressable)
80H	P0	Port 0	bit addressable: P0.7 -> P0.0
81H	SP	Stack Pointer	x
82H	DPL	Low byte of DPTR	x
83H	DPH	High byte of DPTR	x
87H	PCON	Power control	x
88H	TCON	Timer control	TF1-TR1-TF0-TR0-IE1-IT1-IE0-IT0
89H	TMOD	Timer mode control	x
8AH	TL0	Timer 0 low byte	x
8BH	TL1	Timer 1 low byte	x
8CH	TH0	Timer 0 high byte	x
8DH	TH1	Timer 1 high byte	x
90H	P1	Parallel port 1	Bit Addressable P1.7 -> P1.0
98H	SCON	Serial control	SM0-SM1-SM2-REN-TB8-RB8-TI -RI
99H	SBUF	Serial buffer	x
A0H	P2	Port 2	Bit addressable: P2.7-P2.0
A8H	IE	Interrupt Enable	EA - x - x -ES -ET1-EX1-ET0-EX0
B0H	P3	Parallel port 3	Bit addressable: P3.7 -> P3.0
B8H	IP	Interrupt priority	x - x - x -PS -PT1-PX1-PT0-PX0
D0H	PSW	Program Status Word	CY -AC -F0 -RS1-RS0-OV -F1 -P
E0H	ACC	Accumulator	ACC.7 -> ACC.0
F0H	B	B register	B.7 -> B.0

Interrupt control register

IE:	EA	Global Interrupt Enable bit. Set to 0 to *disable* ALL interrupts
	ES	Serial interrupt enable: receive interrupts (RI) or transmit (TI)
	ET0, ET1	Enable Timer0/Timer1 Interrupts (when count rolls over to zero)

Timer control and mode registers - 2 timers 0 and 1

TCON:

TF0/TF1	Timer overflow flag timers for Timer0/Timer1
TR0/TR1	Timer run control bit. Set to 1 by software to enable timer ON

TMOD: mode0 - mode1 bits. 2x4-bit nibbles. Timer 1 right 4 bits, Timer 0 lower 4 bits.
These bits are used to control how the timers behave, whether they auto-reload …

mode = 0	13 bit timer
mode = 1	16 bit timer
mode = 2	8 bit auto-reload timer. THx -> TLx on overflow. Used by Serial I/O as bit rate (*32). THx:= OFDh gives 9600bps for 11.059Mhz clock

Serial control register

SCON: SM0-SM1-SM2-REN should be set to 0111 for normal operation

TI	set when the character has been transmitted
RI	set when a character is received

Power control register

PCON:	set to 2 will stop the processor

Addressing Modes:

Rn	Register R0 - R7 of the currently selected register bank.
direct	8-bit address of a location in internal data memory. This could be an internal Data RAM location (0-127) or a SFR.
@Ri	8-bit Data RAM location addressed indirectly via register R0 or R1.
#data	8-bit constant included in instruction.
#data16	16-bit constant included in instruction.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of Program Memory as the first byte of the following instruction.
addr16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within 2K byte Program Memory address space.
rel, relative	Signed (two's complement) 8-bit offset from current PC. Range is -128 to +127 bytes relative to first byte of the next instruction. Used by SJMP and all conditional jumps (eg JZ, JNZ, JB …).
bit	Direct addressed bit in internal Data RAM or SFR.

Arithmetic operations:

		Byte	Cycle	C	OV	AC
ADD A,Rn	Add register to Accumulator	1	1	X	X	X
ADD A,direct	Add direct byte to Accumulator	2	1	X	X	X
ADD A,@Ri	Add indirect RAM to Accumulator	1	1	X	X	X
ADD A,#data	Add immediate data to Accumulator	2	1	X	X	X
ADDC A,Rn	Add register to Acc. with Carry	1	1	X	X	X
ADDC A,direct	Add direct byte to Acc. with Carry	2	1	X	X	X
ADDC A,@Ri	Add indirect RAM to Acc. with Carry	1	1	X	X	X
ADDC A,#data	Add immediate data to Acc. / Carry	2	1	X	X	X
SUBB A,Rn	Subtract reg. from Acc. with borrow	1	1	X	X	X
SUBB A,direct	Sub. direct byte from Acc. / borrow	2	1	X	X	X
SUBB A,@Ri	Sub. indirect RAM from Acc./ borrow	1	1	X	X	X
SUBB A,#data	Sub. imm. data from Acc. / borrow	2	1	X	X	X
INC A	Increment Accumulator	1	1			
INC Rn	Increment register	1	1			
INC direct	Increment direct byte	2	1			
INC @Ri	Increment indirect RAM	1	1			
DEC A	Decrement Accumulator	1	1			
DEC Rn	Decrement register	1	1			
DEC direct	Decrement direct byte	2	1			
DEC @Ri	Decrement indirect RAM	1	1			
INC DPTR	Increment Data Pointer	1	2			
MUL AB	Multiply A and B	1	4	0	X	
DIV AB	Divide A by B	1	4	0	X	
DA A	Decimal adjust Accumulator used for Binary Coded Decimal adjustment	1	1	X		

Logical operations

		Byte	Cycle	C OV AC
ANL A,Rn	AND register to Accumulator	1	1	
ANL A,direct	AND direct byte to Accumulator	2	1	
ANL A,@Ri	AND indirect RAM to Accumulator	1	1	
ANL A,#data	AND immediate data to Accumulator	2	1	
ANL direct,A	AND Accumulator to direct byte	2	1	
ANL direct,#data	AND immediate data to direct byte	3	2	
ORL A,Rn	OR register to Accumulator	1	1	
ORL A,direct	OR direct byte to Accumulator	2	1	
ORL A,@Ri	OR indirect RAM to Accumulator	1	1	
ORL A,#data	OR immediate data to Accumulator	2	1	
ORL direct,A	OR Accumulator to direct byte	2	1	
ORL direct,#data	OR immediate data to direct byte	3	2	
XRL A,Rn	Exc-OR register to Accumulator	1	1	
XRL A,direct	Exc-OR direct byte to Accumulator	2	2	
XRL A,@Ri	Exc-OR indirect RAM to Accumulator	1	1	
XRL A,#data	Exc-OR immediate data to Acc.	2	1	
XRL direct,A	Exc-OR Accumulator to direct byte	2	1	
XRL direct,#data	Exc-OR imm. data to direct byte	3	2	
CLR A	Clear Accumulator	1	1	
CPL A	Complement Accumulator	1	1	
RL A	Rotate Accumulator left	1	1	
RLC A	Rotate Acc. left through Carry	1	1	X
RR A	Rotate Accumulator right	1	1	
RRC A	Rotate Acc. right through Carry	1	1	X
SWAP A	Swap upper & lower 4 bits in Acc	1	1	

Data transfer

		Byte	Cycle	C OV AC
MOV A,Rn	Move register to Accumulator	1	1	
MOV A,direct	Move direct byte to Accumulator	2	1	
MOV A,@Ri	Move indirect RAM to Accumulator	1	1	
MOV A,#data	Move immediate data to Accumulator	2	1	
MOV Rn,A	Move Accumulator to register	1	1	
MOV Rn,direct	Move direct byte to register	2	2	
MOV Rn,#data	Move immediate data to register	2	1	
MOV direct,A	Move Accumulator to direct byte	2	1	
MOV direct,Rn	Move register to direct byte	2	2	
MOV direct,direct	Move direct byte to direct byte	3	2	
MOV direct,@Ri	Move indirect RAM to direct byte	2	2	
MOV direct,#data	Move immediate data to direct byte	3	2	
MOV @Ri,A	Move Accumulator to indirect RAM	1	1	
MOV @Ri,direct	Move direct byte to indirect RAM	2	2	
MOV @Ri,#data	Move immediate data to indirect RAM	2	1	
MOV DPTR,#data16	Load Data Pointer with 16-bit const	3	2	
MOVC A,@A+DPTR	Move Code byte rel. to DPTR to Acc.	1	2	
MOVC A,@A+PC	Move Code byte rel. to PC to Acc.	1	2	
PUSH direct	Push direct byte onto stack	2	2	
POP direct	Pop direct byte from stack	2	2	
XCH A,Rn	Exchange register with Accumulator	1	1	
XCH A,direct	Exchange direct byte with Acc.	2	1	
XCH A,@Ri	Exchange indirect RAM with Acc.	1	1	
XCHD A,@Ri	Exchange low order digit indirect			
	RAM with Accumulator	1	1	

Number and String Formats

Numbers :	Decimal (default)	: e.g.	34, 127, 255, 0, -1, -27
	Binary	: e.g.	01110101B
	Hexadecimal	: e.g.	$7F, 7Fh, 0FFH, $FF, 0A8H
			note the leading $ or a trailing h or H.

Note: if not preceded by **$** hex constants must start with 0-9. eg **0**C7h

Characters: 'A' - 'Abc' - ‘A’,00DH,00AH (mixed mode), "T"

Strings : 'abc' or "abc". Only with *DB* directive for putting strings into ***CODE memory.*** Use the ***MOVC A,@A+DPTR***, or ***MOVC A,@A+PC*** to access values

Operators : () + - / * MOD SHR SHL NOT AND OR XOR

Boolean variable manipulation

		Byte	Cycle	C OV AC
CLR C	Clear Carry	1	1	0
CLR bit	Clear direct bit	2	1	
SETB C	Set Carry	1	1	1
SETB bit	Set direct bit	2	1	
CPL C	Complement Carry	1	1	X
CPL bit	Complement direct bit	2	1	
ANL C,bit	AND direct bit to Carry	2	2	X
ANL C,/bit	AND complement of dir. bit to Carry	2	2	X
ORL C,bit	OR direct bit to Carry	2	2	X
ORL C,/bit	OR complement of dir. bit to Carry	2	2	X
MOV C,bit	Move direct bit to Carry	2	1	X
MOV bit,C	Move Carry to direct bit	2	2	
JC rel	Jump if Carry is set	2	2	
JNC rel	Jump if Carry not set	2	2	
JB bit,relative	Jump if direct bit is set	3	2	
JNB bit, relative	Jump if direct bit is not set	3	2	
JBC bit, relative	Jump if dir. bit is set & clear bit	3	2	

Program Branching

		Byte	Cycle	C OV AC
ACALL addr11	Absolute subroutine call	2	2	
LCALL addr16	Long subroutine call	3	2	
RET	Return from subroutine	1	2	
RETI	Return from interrupt	1	2	
AJMP addr11	Absolute jump (dest in same 2K page)	2	2	
LJMP addr16	Long jump - jump anywhere (safest)	3	2	
SJMP rel	Short jump (relative address)	2	2	
JMP @A+DPTR	Jump indirect relative to the DPTR	1	2	
JZ rel	Jump if Accumulator is zero	2	2	
JNZ rel	Jump if Accumulator is not zero	2	2	
CJNE A,direct,rel	Compare direct byte to Accumulator and jump if not equal	3	2	X
CJNE A,#data,rel	Compare immediate data to Accumulator and jump if not equal	3	2	X
CJNE Rn,#data,rel	Compare immediate data to register and jump if not equal	3	2	X
CJNE @Ri,#data,rel	Compare immediate data to indirect RAM and jump if not equal	3	2	X
DJNZ Rn,rel	Decrement register, jump if not zero	2	2	
DJNZ direct,rel	Decrement direct byte and jump if not zero	3	2	
NOP	No operation	1	1	

Assembler Directives and Controls

```
;                       Everything after a semicolon (;) on the same line is a comment
Label:                  Must start in column 1 - Defines a new Label - colon is optional.
```

Controlling Memory Spaces and Code location

```
            ORG   56H   Specify a value for the current segment's location counter.
            USE   IRAM  Makes the data space the currently selected segment
            USE   ROM   Makes the code space the currently selected segment
```

Defining Byte and Bit values

```
TEN         EQU   10    EQUates 10 to symbol TEN, like #define in C, CONST in Delphi
ON_FLAG     BIT   6     Assigns BIT 6 (in data or SFR space) to the symbol ON_FLAG
```

Allocating Memory

```
SP_BUFFER:  RMB   6     Reserves Memory Byte - reserves 6 bytes of storage in current
                        memory space (affected by most recent USE IRAM/ROM).
Message:    DB    'Hi'  Define Byte(s): Store byte constants in code space.

; The following are all equivalent - the string hello followed by a newline and a null.
newline     EQU   13
            DB    "H","E","L","L","O",13,0
            DB    "Hello",13,0
            DB    "Hello",newline,0
```