SHORT ARCHITECTURE of 8051-type MICROCOMPUTER

MEMORY

Program memory

0000...0FFF 4 kbyte ROM, EPROM, FLASH
0000...1FFF 8 kbyte ROM, EPROM, FLASH
0000...1FFF 16 kbyte ROM, EPROM, FLASH
0000...1FFF 32 kbyte ROM, EPROM, FLASH
0000...FFFF 64 kbyte ROM, EPROM, FLASH

Read-write memory

00...7F random access memory
80...FF special function registers

Registers

ident	name	address in RAM	primary use	after RESET
A	accumulator	E0	destination and source for arithm. & logic. operation	00
B	B-register	F0	for hardware multiplication and division	00
R0	register 0	00 (08, 10, 18)	scratchpad register, indirect (@) addressing	xx
R1	register 1	01 (09, 11, 19)	scratchpad register, indirect (@) addressing	xx
R2	register 2	02 (0A, 12, 1A)	scratchpad register	xx
R3	register 3	03 (0B, 13, 1B)	scratchpad register	xx
R4	register 4	04 (0C, 14, 1C)	scratchpad register	xx
R5	register 5	05 (0D, 15, 1D)	scratchpad register	xx
R6	register 6	06 (0E, 16, 1E)	scratchpad register	xx
R7	register 7	07 (0F, 17, 1F)	scratchpad register	xx
P0	I/O port 0	80	8-bit input / ouput port, data / address for ext. mem.	FF
P1	I/O port 1	90	8-bit input / ouput port	FF
P2	I/O port 2	A0	8-bit input / ouput port, address output for ext. mem.	FF
P3	I/O port 3	B0	8-bit input / ouput port	FF
PSW	program status word	D0	Carry (bit 7), Overflow (bit 2), Parity (bit 0), Register bank selection (bit 3 & 4)	00
SP	stack pointer	81	points to the latest used position in the stack	07

STACK

The stack is used for short time storage of variables (PUSH and POP) and for storage of return addresses for subroutines (CALL and RET). In the default case this information is stored in RAM addresses 08, 09, 0A and up. If you want to use this memory space, change the stack pointer SP to e.g. 2F in the beginning (initialisation) of your program.

Hardware and software descriptions

All information on the hardware and software is contained in the official descriptions, which can be found at:
http://www.nxp.com/acrobat_download/various/80C51_FAM_ARCH_1.pdf
http://www.nxp.com/acrobat_download/various/80C51_FAM_HARDWARE_1.pdf
http://www.nxp.com/acrobat_download/various/80C51_FAM_PROG_GUIDE_1.pdf

INSTRUCTIONS and OPERATIONS

Arithmetic and Logical

			<destination>	<source>
ADD	add <dest> , <src>	<dest> ← <dest> + <src>	A	A, R_n, @, #, M
ADDC	add with carry	<dest> ← <dest> + <src> + carry bit	A	A, R_n, @, #, M
SUBB	subtract with borrow	<dest> ← <dest> – <src> – carry bit	A	A, R_n, @, #, M
INC	increment	<dest> ← <dest> + 1	A, R_n, @, M	
DEC	decrement	<dest> ← <dest> – 1	A, R_n, @, M	
ANL	logic AND	<dest> ← <dest> bit wise AND <src>	A ↔	A, R_n, @, #, M
ORL	logic OR	<dest> ← <dest> bit wise OR <src>	A ↔	A, R_n, @, #, M
XRL	logic exclusive or	<dest> ← <dest> bit wise excl. or <src>	A ↔	A, R_n, @, #, M
MUL	multiply (only A and B)	B A ← A x B		
DIV	divide (only A and B)	A ← A / B and B ← remainder of A / B		
CLR	clear	<dest> ← 0	A, carry, bit	
CPL	complement	<dest> ← <dest> bit wise inverted (NOT)	A, carry, bit	
RL	rotate left	shift bits to the left, MSB into LSB	A	
RLC	rotate left via carry	shift bits left, carry into LSB, MSB into carry	A	
RR	rotate right	shift bits to the right, LSB into MSB	A	
RRC	rotate right via carry	shift bits rignt, carry into MSB, LSB to carry	A	
DA	decimal adjust	for additions in Binary Coded Decimal	A	
SWAP	swap (4 bits) nibbles	swap bits 7...4 with bits 3...0	A	
SETB	set bit	<bit> ← 1	carry, bit	

Data transfer

			<destination>	<source>
MOV	move <dest> , <src>	<dest> ← <src>	A, R_n, @, M	A, R_n, @, #, M
XCH	exchange	<dest> ↔ <src>	A	A, R_n, @, M
XCHD	excange decimal	<dest> ↔ <src> only for bits 3..0	A	@
PUSH	push on stack	<src> → stack	stack	M
POP	pop from stack	<dest> ← stack	M	stack

Program branching

			<destination>	<source>
AJMP	absolute jump	jump to 11-bits address in program		
LJMP	long jump	jump to 16-bits address in program		
SJMP	short (relative) jump	jump to ± 127 bytes in program		
NOP	no operation	do nothing		
JZ	jump if zero	jump if A is 00		A
JNZ	jump if not zero	jump if A is not 00		A
JC	jump if carry	jump if carry bit is 1		carry
JNC	jump if no carry	jump if carry bit is 0		carry
JNB	jump if no bit	jump if <bit> is 0		bit
JBC	jump if bit and clear	jump if <bit> is 1, and make <bit> 0		bit
ACALL	absolute call subroutine	addr. → stack & jump to 11-bit address	stack	old progr.addr.
LCALL	long call subroutine	addr. → stack & jump to 16-bit address	stack	old progr.addr.
RET	return from subroutine	jump to address from stack	progr. addr.	stack
RETI	return from interrupt	return at the end of interrupt routine	progr. addr.	stack
CJNE	compare, jump not equal	jump if <dest> ≠ <src>	A, R_n, @	#, M
DJNZ	decrement, jump not zero	<dest> ← <dest> – 1, jump if <dest> ≠ 00	R_n, M	

ARITHMETIC and LOGICAL OPERATIONS

	ADD A,	ADDC A,	SUBB A,	INC	DEC	ANL A,	ANL addr	ORL A,	ORL addr	XRL A,	XRL addr
A				04	14		A2 aa		42 aa		62 aa
R0	28	38	98	08	18	58		48		68	
R1	29	39	99	09	19	59		49		69	
R2	2A	3A	9A	0A	1A	5A		4A		6A	
R3	2B	3B	9B	0B	1B	5B		4B		6B	
R4	2C	3C	9C	0C	1C	5C		4C		6C	
R5	2D	3D	9D	0D	1D	5D		4D		6D	
R6	2E	3E	9E	0E	1E	5E		4E		6E	
R7	2F	3F	9F	0F	1F	5F		4F		6F	
@R0	26	36	96	06	16	56		46		66	
@R1	27	37	97	07	17	57		47		67	
#data	24 nn	34 nn	94 nn			54 nn	53 aa nn	44 nn	43 aa nn	64 nn	63 aa nn
address	25 aa	35 aa	95 aa	05 aa	15 aa	55 aa		45 aa		65 aa	

MUL AB	A4
DIV AB	84

	CLR	CPL	RL	RLC	RR	RRC	DA	SWAP
A	E4	F4	23	33	03	13	D4	C4

BIT MANIPULATION

	CLR	CPL	SETB	ANL C,	ORL C,	MOV C,	MOV bit
C	C3	B3	D3				92 rb
bit	C2 rb	B2 rb	D2 rb	82 rb	72 rb	A2 rb	
/bit				B0 rb	A0 rb		

aa = 1-byte address
nn = 1-byte numerical
rb = register bit address

DATA TRANSFER

	MOV A,	MOV R0,	MOV R1,	MOV R2,	MOV R3,	MOV R4,	MOV R5,	MOV R6,	MOV R7,	MOV @R0,	MOV @R1,	MOV addr	XCH A,	XCHD A,
A		F8	F9	FA	FB	FC	FD	FE	FF	F6	F7	F5 aa		
R0	E8											88	C8	
R1	E9											89	C9	
R2	EA											8A	CA	
R3	EB											8B	CB	
R4	EC											8C	C8	
R5	ED											8D	C8	
R6	EE											8E	CE	
R7	EF											8F	CF	
@R0	E6											86 aa	C6	B6
@R1	E7											87 aa	C7	B7
#data	74 nn	78 nn	79 nn	7A nn	7B nn	7C nn	7D nn	7E nn	7F nn	76 nn	77 nn	75 aa nn		
address	E5 aa	A8 aa	A9 aa	AA aa	AB aa	AC aa	AD aa	AE aa	AF aa	A6 aa	A7 aa	85 from to	C5 aa	

	MOV DPTR,	MOVC A,
#data16	90 nnh nnl	
@A+DPTR		93
@A+PC		83

	MOVX A,	MOVX @P0	MOVX @R1	MOVX @DPTR
A		F2	F3	F0
@R0	E2			
@R1	E3			
@DPTR	E0			

PUSH addr	POP addr
C0 aa	D0 aa

aa = 1-byte address
nn = 1-byte numerical
nnh = high byte of 16-bit numerical
nnl = low byte of 16-bit numerical

PROGRAM BRANCHING

Unconditional Jumps											
AJMP								LJMP	SJMP	JMP @A,DPTR	NOP
00aa	01aa	02aa	03aa	04aa	05aa	06aa	07aa				
01 aa	21 aa	41 aa	61 aa	81 aa	A1 aa	C1 aa	E1 aa	02 aah aal	80 ±d	73	00

Conditional Jumps						
JZ	JNZ	JC	JNC	JB	JNB	JBC
60 ±d	70 ±d	40 ±d	50 ±d	20 rb ±d	30 rb ±d	10 rb ±d

aa = 1-byte address
aah = high byte of 16-bit address
aal = low byte of 16-bit address
nn = 1-byte numerical
rb = register bit address
±d = relative displacement
7F = +127, 80 = -128

Subroutines										
ACALL								LCALL	RET	RETI
00aa	01aa	02aa	03aa	04aa	05aa	06aa	07aa			
11 aa	31 aa	51 aa	71 aa	91 aa	B1 aa	D1 aa	F1 aa	12 aah aal	22	32

	Compare and Jump if Not Equal										
	CJNE A,	CJNE R0,	CJNE R1,	CJNE R2,	CJNE R3,	CJNE R4,	CJNE R5,	CJNE R6,	CJNE R7,	CJNE @R0,	CJNE @R1,
#data, rel	B4 nn ±d	B8 nn ±d	B9 nn ±d	BA nn ±d	BB nn ±d	BC nn ±d	BD nn ±d	BE nn ±d	BF nn ±d	B6 nn ±d	B7 nn ±d
address, rel	B5 nn ±d										

Decrement and Jump if Not Zero								
DJNZ address,	DJNZ R0,	DJNZ R1,	DJNZ R2,	DJNZ R3,	DJNZ R4,	DJNZ R5,	DJNZ R6,	DJNZ R7,
D5 aa ±d	D8 ±d	D9 ±d	DA ±d	DB ±d	DC ±d	DD ±d	DE ±d	DF ±d