I. Instruction Set

Instruction Set	Instruction	B	C	Opcode		
AND -> A	ANL A, Rn	1	1	01011111		
	ANL A, direct	2	1	01010101	<direct>	
	ANL A, @Ri	1	1	0101011i		
	ANL A, #<data>	2	1	01010100	<data>	
	ANL direct, A	2	1	01010010	<direct>	
	ANL direct, #<data>	3	2	01010011	<direct>	<data>
OR -> A	ORL A, Rn	1	1	01001111		
	ORL A, direct	2	1	01000101	<direct>	
	ORL A, @Ri	1	1	0100011i		
	ORL A, #<data>	2	1	01000100	<data>	
	ORL direct, A	2	1	01000010	<direct>	
	ORL direct, #<data>	3	2	01000011	<direct>	<data>
XOR -> A	XRL A, Rn	1	1	01101nnn		
	XRL A, direct	2	1	01100101	<direct>	
	XRL A, @Ri	1	1	0110011i		
	XRL A, #<data>	2	1	01100100	<data>	
	XRL direct, A	2	1	01100010	<direct>	
	XRL direct, #<data>	3	2	01100011	<direct>	<data>
rotate ACC right	RR A	1	1	00000011		
rotate ACC left	RL A	1	1	00100011		
rotate ACC right through C	RRC A	1	1	00010011		
rotate ACC left through C	RLC A	1	1	00110011		
swap low and high nibbles in ACC	SWAP	1	1	11000100		
set bit low	CLR C	1	1	11000011		
	CLR <bit>	2	1	11000010	<bit address>	
complement bit	CPL C	1	1	10110011		
	CPL <bit>	2	1	10110100	<bit address>	
set bit high	SETB C	1	1	11010011		
	SETB <bit>	2	1	11010010	<bit address>	
single-bit logic	ANL C, <bit>	2	2	10000010	<bit address>	
	ANL C, /<bit>	2	2	10110000	<bit address>	
	ORL C, <bit>	2	2	01110010	<bit address>	
	ORL C, /<bit>	2	2	10100000	<bit address>	
bitwise move	MOV C, <bit>	2	1	10100010	<bit address>	
	MOV <bit>, C	2	2	10010010	<bit address>	
Jump if Carry set	JC <offset>	2	2	01000000	<offset>	
Jump if Carry clear	JNC <offset>	2	2	01010000	<offset>	
Jump if bit set	JB <bit>, <offset>	3	2	00100000	<bit address>	<offset>
Jump if bit clear	JNB <bit>, <offset>	3	2	00110000	<bit address>	<offset>
Jump and clear bit if bit set	JBC <bit>, <offset>	3	2	00010000	<bit address>	<offset>
A + val -> A	ADD A, Rn	1	1	00101nn		
	ADD A, <direct>	2	1	00100101	<direct>	
	ADD A, @Ri	1	1	0010011i		
	ADD A, #<data>	2	1	00100100	<data>	
A + val + C -> A	ADDC A, Rn	1	1	00111nnn		
	ADDC A, <direct>	2	1	00110101	<direct>	
	ADDC A, @Ri	1	1	0011011i		
	ADDC A, #<data>	2	1	00110100	<data>	
A - val -> A	SUBB A, Rn	1	1	10011nnn		
	SUBB A, <direct>	2	1	10010101	<direct>	
	SUBB A, @Ri	1	1	1001011i		
	SUBB A, #<data>	2	1	10010100	<data>	
Increment	INC A	1	1	00000100		
	INC Rn	1	1	00001nnn		
	INC <direct>	2	1	00000101	<direct>	
	INC @Ri	1	1	0000011i		
	INC DPTR	1	2	10100011		
Decrement	DEC A	1	1	00010100		
	DEC Rn	1	1	00011nnn		
	DEC <direct>	2	1	00010101	<direct>	
	DEC @Ri	1	1	0001011i		
Multiply : low -> A, high -> B	MUL AB	1	4	10100100		
A/B -> A, A%B -> B	DIV AB	1	4	10000100		
dest <- src	MOV @Ri, #<data>	2	1	0111011i	<data>	
	MOV @Ri, A	1	1	1111011i		
	MOV @Ri, <direct>	2	2	1010011i	<src>	
	MOV A, #<data>	2	1	01110100	<data>	
	MOV A, @Ri	1	1	1110011i		
	MOV A, <direct>	2	1	11100101	<src>	
	MOV A, Rn	1	1	11101nnn		
	MOV <direct>, <direct>	3	2	10000101	<dest>	<src>
	MOV <direct>, #<data>	3	2	01110101	<dest>	<data>
	MOV <direct>, @Ri	2	2	1000011i	<dest>	
	MOV <direct>, A	2	1	11110101	<dest>	
	MOV <direct>, Rn	2	2	10001nnn	<dest>	
	MOV DPTR, #<data>	3	2	10010000	<data|high>	<data|low>
	MOV Rn, #<data>	2	1	01111nnn	<data>	
	MOV Rn, A	1	1	11111nnn		
	MOV Rn, <direct>	2	2	10101nnn	<src>	
no fucking clue	MOVC A, @A+DPTR	1	2	10010011		
	MOVC A, @A+PC	1	2	10000011		
ACC <-> ext. memory	MOVX @Ri, A	1	2	1111001i		
	MOVX A, @DPTR	1	2	11100000		
	MOVX A, @Ri	1	2	1110001i		
stack operations	PUSH <direct>	2	2	11000000	<src>	
	POP <direct>	2	2	11010000	<dest>	
exchange A and src	XCH A, @Ri	1	1	1100011i		
	XCH A, <direct>	2	1	11000101	<src>	
	XCH A, Rn	1	1	11001nnn		
exchange low nibble	XCHD A, @Ri	1	1	1101011i		
calling subroutines	ACALL <direct(11b)>	2	2	aaa10001	<addr|low>	
	LCALL <direct(16b)>	3	2	00010010	<addr|high>	<addr|low>
	RET	1	2	00100010		
	RETI	1	2	00110010		
jumping and branching	AJMP <direct(11b)>	2	2	aaa00001	<addr|low>	
	LJMP <direct(16b)>	3	2	00000010	<addr|high>	<addr|low>
	SJMP <offset(8b)>	2	2	10000000	<offset>	
	JMP @A+DPTR	1	2	01110011		
	JZ <offset(8b)>	2	2	01100000	<offset>	
	JNZ <offset(8b)>	2	2	01110000	<offset>	
	CJNE @Ri, #<data>, <offset(8b)>	3	2	1011011i	<data>	<offset>
	CJNE A, #<data>, <offset(8b)	3	2	10110100	<data>	<offset>
	CJNE A, <direct>, <offset(8b)	3	2	10110101	<src>	<offset>
	CJNE Rn, #<data>, <offset(8b)	3	2	10111nnn	<data>	<offset>
	DJNZ <direct>, <offset(8b)>	3	2	11010101	<src>	<offset>
	DJNZ Rn, <offset(8b)>	2	2	11011nnn	<offset>	
do nothing	NOP	1	1	00000000		

II. 8051 Organization

ROM	4K
RAM	128B
Address Bus	16b
I/O	4x 8b parallel ports, 1x serial port (all bit-addressable)

A. Components

whatthefuck.png

1) Embedded System Advantages:

- low power consumption
- small size
- rugged operating ranges
- low unit cost

B. Program Flow

Instruction Fetch Cycle	1	program & data read in and stored in main memory
	2	Instruction fetched from main memory
	3	Fetched instruction is decoded and investigated
Instruction Execution Cycle	4	Operand(s) fetched
	5	Operation performed
	6	GOTO 1

C. Addressing Modes

Implied Mode	operands are specified implicitly
Immediate Mode	operands evaluate to values
Index Mode	effective address determined by both the contents of the index register and a displacement
Register Mode	operand is a register containing wanted value
Direct Mode	operands are addresses pointing to wanted value
Indirect Register Mode	operand is either R0 or R1, which contains an address pointing to wanted value

III. I/O

A. Serial Port Operation

B. Timer Operation

IV. Examples