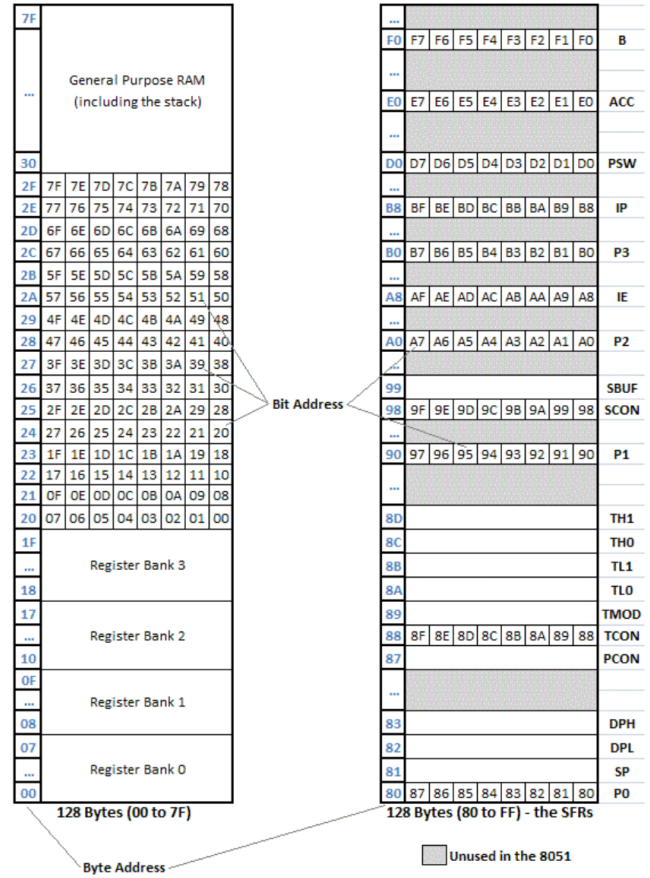


## I. Instruction Set

Instruction Set	Instruction	B	C	Opcode	Operand 1	Operand 2
AND -> A	ANL A, Rn	1	1	01011111	<direct>	
	ANL A, direct	2	1	01010101	<direct>	
	ANL A, @Ri	1	1	01010111	<data>	
	ANL A, #<data>	2	1	01010100	<data>	
	ANL direct, A	2	1	01010010	<direct>	<data>
OR -> A	ORL A, Rn	1	1	01011111	<direct>	
	ORL A, direct	2	1	01000101	<direct>	
	ORL A, @Ri	1	1	01000111	<data>	
	ORL A, #<data>	2	1	01000100	<data>	
	ORL direct, A	2	1	01000010	<direct>	<data>
XOR -> A	XRL A, Rn	1	1	01101111	<direct>	
	XRL A, direct	2	1	01100101	<direct>	
	XRL A, @Ri	1	1	01100111	<data>	
	XRL A, #<data>	2	1	01100100	<data>	
	XRL direct, A	2	1	01100010	<direct>	<data>
rotate ACC right	RR A	1	1	00000011		
	rotate ACC left	1	1	00100011		
	rotate ACC right through C	1	1	00010011		
	rotate ACC left through C	1	1	00110011		
	swap low and high nibbles in ACC	1	1	10001000		
set bit low	CLR C	1	1	11000011	<bit address>	
	CPL C	1	1	10110011	<bit address>	
	CPL, <bit>	2	1	10110100	<bit address>	
	SETB C	1	1	11010011	<bit address>	
	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	01010000	<bit address>	
	MOV C, <bit>	1	1	11010001	<bit address>	
hitwise move	MOV, <bit>, C	2	2	10010010	<bit address>	
	JC, <offset>	2	2	01000000	<offset>	
	JNC, <offset>	2	2	01010000	<offset>	
	JB, <bit>, <offset>	3	2	00100000	<bit address>	<offset>
	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>
	ADD A, Rn	1	1	00101111	<direct>	
	ADD A, @Ri	1	1	00100111	<data>	
	ADD A, #<data>	2	1	00100100	<data>	
	ADDC A, Rn	1	1	00111111	<direct>	
A + val -> A	ADDC A, <direct>	2	1	00110101	<direct>	
	ADDC A, @Ri	1	1	00110111	<data>	
	ADDC A, #<data>	2	1	00110100	<data>	
	SUBB A, Rn	1	1	10011111	<direct>	
	SUBB A, @Ri	1	1	10010111	<data>	
A - val -> A	SUBB A, #<data>	2	1	10010100	<data>	
	INC A	1	1	00000100		
	INC Rn	1	1	00001111	<direct>	
	INC, <direct>	2	1	00000101	<direct>	
	INC @Ri	1	1	00000111	<data>	
Increment	INC DPTR	1	2	10100011		
	DEC A	1	1	00010100		
	DEC Rn	1	1	00011111	<direct>	
	DEC, <direct>	2	1	00010101	<direct>	
	DEC @Ri	1	1	00010111	<data>	
Decrement	MUL AB	1	4	10101000		
	DIV AB	1	4	10001000		
	MOV @Ri, #<data>	2	1	01110111	<data>	
	MOV @Ri, A	1	1	11110111	<src>	
	MOV @Ri, <direct>	2	1	10100111	<data>	
dest <- src	MOV A, #<data>	2	1	01110100	<data>	
	MOV A, @Ri	1	1	11100111	<src>	
	MOV A, <direct>	2	1	11100101	<src>	
	MOV A, Rn	1	1	11101111	<src>	
	MOV, <direct>, <direct>	1	1	11101100	<dest>	<src>
no fucking clue	MOV, <direct>, #<data>	3	2	01110101	<dest>	<data>
	MOV, <direct>, @Ri	1	1	10000111	<dest>	<data>
	MOV, <direct>, A	2	1	11110101	<dest>	<data>
	MOV, <direct>, Rn	2	2	10001111	<dest>	<data>
	MOV DPTR, #<data>	2	2	10010000	<data>	<data>
stack operations	MOV Rn, #<data>	2	1	11111111	<data>	
	MOV Rn, A	1	1	11111110	<data>	
	MOV Rn, <direct>	2	2	10101111	<src>	
	MOV Rn, @Ri	1	1	11001111	<src>	
	MOV Rn, #<data>	2	2	11011111	<src>	
exchange A and src	MOVC A, @A+DPTR	1	2	10010011		
	MOVC A, @A+PC	1	2	10000011		
	MOVX @Ri, A	1	2	11110011		
	MOVX A, @DPTR	1	2	11100000		
	MOVX A, @Ri	1	2	11100011		
ACC <-> ext. memory	PUSH <direct>	2	2	11000000	<src>	
	POP <direct>	2	2	11000000	<dest>	
	XCH A, @Ri	1	1	11000111	<src>	
	XCH A, <direct>	2	1	11000101	<src>	
	XCH A, Rn	1	1	11001111	<src>	
exchange low nibble	XCHD A, @Ri	1	1	11010111	<src>	
	ACALL <direct(11b)>	2	2	aaa10001	<addr/low>	<addr/high>
	LCALL <direct(16b)>	3	2	00010010	<addr/low>	<addr/high>
	RET	1	2	00100010	<addr/low>	<addr/high>
	RETI	1	2	00110010	<addr/low>	<addr/high>
calling subroutines	AJMP <direct(11b)>	2	2	aaa00001	<addr/low>	<addr/high>
	LJMP <direct(16b)>	3	2	00000010	<addr/low>	<addr/high>
	SJMP <offset(8b)>	2	2	10000000	<offset>	
	JMP @A+DPTR	1	2	01110011	<offset>	
	JZ <offset(8b)>	2	2	01100000	<offset>	
jumping and branching	JNC <offset(8b)>	2	2	01110000	<offset>	
	CJNE @Ri, #<data>, <offset(8b)>	3	2	10110111	<data>	<offset>
	CJNE A, #<data>, <offset(8b)>	3	2	10110100	<data>	<offset>
	CJNE A, <direct>, <offset(8b)>	3	2	10110101	<data>	<offset>
	CJNE Rn, #<data>, <offset(8b)>	3	2	10111111	<data>	<offset>
do nothing	DJNZ <direct>, <offset(8b)>	3	2	11010101	<src>	<offset>
	DJNZ Rn, <offset(8b)>	2	2	11011111	<src>	<offset>
	NOP	1	1	00000000		

## B. Memory Organization



## II. 8051 Organization

Programs can be in either "absolute form", or "relocatable form". If they are in relocatable form, linking is required to set the absolute addresses for execution.

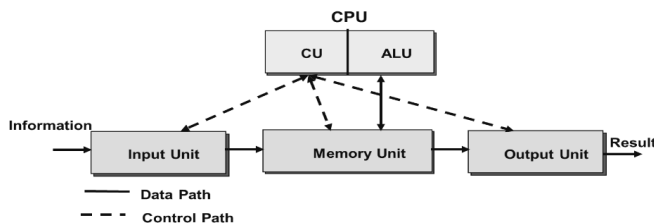
ROM	4K
RAM	128B
Address Bus	16b
I/O	4x 8b parallel ports, 1x serial port (all bit-addressable)

PSW.7	CY	0xD7
PSW.6	AC	0xD6
PSW.5	F0	0xD5
PSW.4	RS1	0xD4
PSW.3	RS0	0xD3
PSW.2	OV	0xD2
PSW.1	0xD1	
PSW.0	P	0xD0

## C. Program Flow

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

## A. Components



### 1) Embedded System Advantages:

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

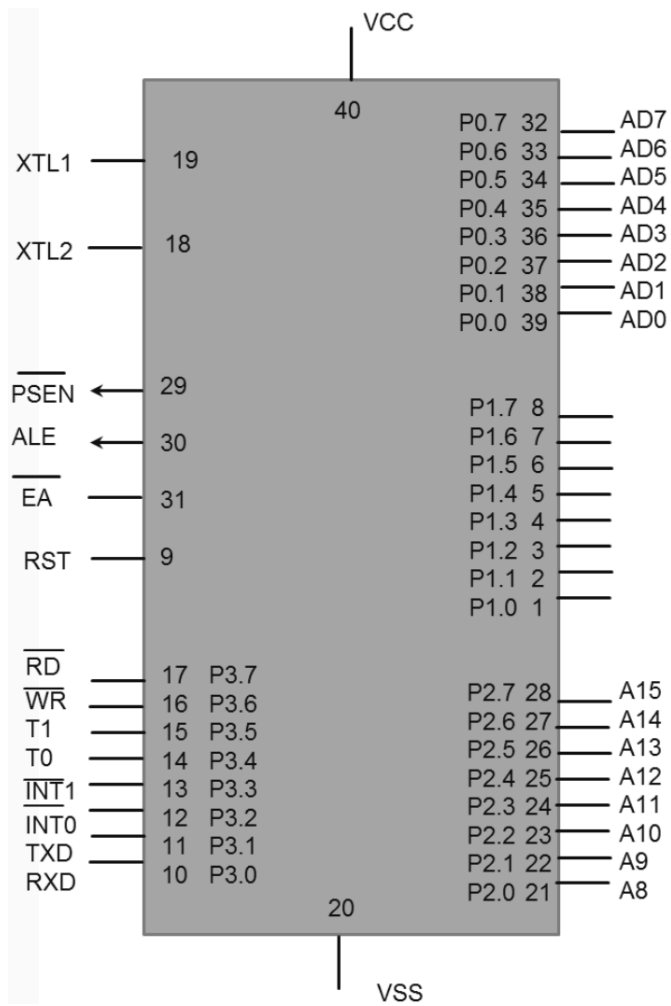
## D. Addressing Modes

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

## IV. Examples

## A. Pinout



## B. Serial Port Operation

## C. Timer Operation