I. Instruction Set

SETB C	ANL A, Ra ANL A, direct	2 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1010101	<data></data>
ANY, direct, # d	ANL direct: A data> OR> A O	1 1 1 0 2 1 1 0 2 1 1 0 3 2 0 1 1 1 1 0 2 1 1 0 3 2 1 1 0 3 1 2 1 1 0 3 1 2 1 1 0 3 1 2 1 1 0 3 1 2 1 1 0 3 1 2 1 1 0 3 1 2 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1	1010111	
ARL direct, # dedata 1 1 0000001 000000 0000000 0000000 0000000 000000	ANL direct: A data> OR> A O	2 1 1 0 2 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1	1010100	
ARL direct, # dedata 1 1 0000001 000000 0000000 0000000 0000000 000000	ANL direct: A data> OR> A O	2 1 1 0 3 2 1 0 1 1 1 0 1 1 1 0 2 1 1 0 2 1 1 0 3 2 1 0 3 1 1 1 0 3 1 2 1 0 4 1 1 1 0 3 1 2 1 1 0 4 1 1 1 0 4 1 1 1 1 0 4 1 1 1 1 0 4 1 1 1 1 1 0 4 1 1 1 1 1 1 1 1 4 1 1 1 1 1 1 1 1 4 1 1 1 1	1010010 <direct> 1010011 <direct> 1010011 <direct> 100111 1001111 <direct> 1000101 <direct> 1000101 <direct> 1000010 <direct> 1000010 <direct> 1010101 <direct> 1100101 <direct> 1100101 <direct> 1100101 <direct> 1100101 <direct> 1100101 <direct> 1100100 <direct> 110010</direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct></direct>	
OR. > A OR. A. firect OR. A. gettal OR. G. gettal OR. A.	ORL A. Recet	1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	100111	
OR > A OR A direct OR A direct OR Collect C	OR -> A	2 1 0 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1	1000101 direct >	<data></data>
ORL A. 68th 1 1 1 1 1 1 1 1 1	OR -> A	1 1 0 2 1 0 0 2 1 1 0 0 2 1 1 0 0 0 1 1 1 1	1000111 10001100	<data></data>
ORL direct, #\ \text{Adatas}	ORL direct, A ORL direct,	2 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1	1000010	<data></data>
ORL direct, #Cetaba> 3 2 01000011 cdirect> XRL A. Rect 1 1 0100111 cdirect> XRL A. Rect XRL A. GRI 1 1 0100111 cdirect> XRL A. GRI 1 1 0000011 cdirect> XRL A. GRI XRL A. G	ORL direct, # <data td="" ="" <=""><td>3 2 0 1 1 1 0 2 1 0 2 1 0 2 1 0 3 2 1 0 3 2 1 0 3 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1</td><td>1000011</td><td><data></data></td></data>	3 2 0 1 1 1 0 2 1 0 2 1 0 2 1 0 3 2 1 0 3 2 1 0 3 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1	1000011	<data></data>
NRIA A first 1	XOR -> A	2 1 1 0 1 1 1 0 2 1 1 0 0 2 1 1 0 0 2 1 1 0 0 0 1 1 1 1	1100101 <direct> 110011i 1100100 <data></data></direct>	
NRL A. # Cdata NRL A. # Cdata 1	XOR. > A	1 1 0 0 2 1 0 0 2 1 1 0 0 0 1 1 1 0 0 1 1 1 1	110011i 1100100 <data></data>	
XRL direct, A 2	XRL direct. A XRL direct.	2 1 0 2 1 0 3 2 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1	1100100 <data></data>	
XRL direct, #\ \text{Actacles} \times \text	Totale ACC right	2 1 0 3 2 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 0	1100010	
Mate	ordate ACC right relate ACC right relate ACC light rotate ACC light rotate ACC light through C swap low and high nibbles in ACC SWAP SWAP CHR C CHL C CH	1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 1		
Totale ACC left Ric A	Totate ACC left	1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 1		<data></data>
Totale ACC eight through C RICA 1 1 00110011	Totate ACC first through C	1 1 0 1 1 0 1 1 1		
rotate ACC left through C SWAF see bit low CLR < SWAF CR < SWAF 1 1 10000100 chi address CR CR < 1 1 1 10000100 chi address CR CR < 1 1 1 10000100 chi address CR CR < 1 1 1 10000100 chi address complement bit CR CR < 1 1 1 10000100 chi address complement bit CR CR < 1 1 1 10000100 chi address complement bit CR CR < 1 1 1 10000100 chi address single-bit logic CR CR CR < 1 1 1 10000100 chi address SETB Cr CR < 1 1 1 10000100 chi address SETB Cr CR < 1 1 1 10000100 chi address single-bit logic CR CR CR CR < 1 1 1 10000100 chi address SETB Cr CR < 1 1 1 10000100 chi address chi address SETB Cr CR < 1 1 1 10000100 chi address CR CR CR < 1 1 1 10000100 chi address ch	rotate ACC left through C RLC A wwap Jow and high mibbles in ACC SWAP case thit low CLR C complement bit CPL C CPL C CPL C set bit high SETB C sETB C SETB C ANL C, Cells ORL C, Cells ORL C, Cells ORL C, Cells ORL C, Cells ORL C, Cells bitwise move MOV C, Clui C, Cells Jump II Carry set JC Coffset D Jump Carry clear JRC Coffset D	1 1 0 1 1 1		+
Set bit low	Set bit low			
Set bit low	Set bit low			
CPL cht	CPL cbit			
CPL cht	CPL cbit		0110011	
SETE Chir Sete About	SETE < bit>	2 1 1		
ABL C, Single-bit logic ORL C, Chi> ORL C, ORL C, Chi> ORL C, 	ANL C, < bit>	1 1 1	1010011	
Month Mont	bitwise move	2 2 1	0000010 obit address>	+
Month Mont	bitwise move	2 2 1	0110000 oit address>	1
District	bitwise move	2 2 0	1110010 ott address>	1
Jump Carry velaw JC coffeet> 2 2 01000000 coffeet> Jump Carry velaw JRC coffeet> 2 2 01000000 coffeet> Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jum	Jump if Carry set JC <offset> Jump if Carry clear JNC <offset> Jump if Carry clear JNC <offset> JR </offset></offset></offset>	2 2 1	0100000 bit address>	+
Jump Carry velaw JC coffeet> 2 2 01000000 coffeet> Jump Carry velaw JRC coffeet> 2 2 01000000 coffeet> Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jump Jum	Jump if Carry set JC <offset> Jump if Carry clear JNC <offset> Jump if Carry clear JNC <offset> JR </offset></offset></offset>	2 2 1	0010010 obit address>	1
Jump and clear bit if bit set	Jump if Carry clear JNC <offset> Jump if bit set JB <bi>, <offset></offset></bi></offset>	2 2 0	1000000 <offset></offset>	
Jump and clear bit if bit set	Jump II on Sec JD Colt >, Colliset >	2 2 0	1010000 <offset></offset>	<offset></offset>
A + val -> A A + val -> A A + val + C -> A A DD A, #Gata> A + val + C -> A ADD A, #Cata> A DD A, #Gata> A + val + C -> A ADD A, #Cata> A DD A, #Gata> A - val -> A ADD A, #Cata> A DD A, #Gata> A - val -> A ADD C A, 6Ri ADD C A, 6Ri ADD C A, 6Ri ADD C A, 6Ri BUBB A, #Cata> A - val -> A BUBB A, #Cata> BUBB A, #Cata		3 2 0	0110000 oit address>	
A + val -> A A + val -> A A + val + C -> A A DD A, #Gata> A + val + C -> A ADD A, #Cata> A DD A, #Gata> A + val + C -> A ADD A, #Cata> A DD A, #Gata> A - val -> A ADD A, #Cata> A DD A, #Gata> A - val -> A ADD C A, 6Ri ADD C A, 6Ri ADD C A, 6Ri ADD C A, 6Ri BUBB A, #Cata> A - val -> A BUBB A, #Cata> BUBB A, #Cata	Jump and clear bit if bit set JBC <b< td=""><td>3 2 0</td><td>00100000 bit address></td><td><offset></offset></td></b<>	3 2 0	00100000 bit address>	<offset></offset>
ADD A, #Colata>	ADD A, Rn	1 1 0	0101nn	
ADD A, #Colata>	A + val -> A ADD A GD			
A DBC A Re		2 1 0		
A - val -> A SUBB A, & direct> 1 1 1 1 1 1 1 1 1	ADDC A, Rn	1 1 0	0111nnn	1
A - val -> A SUBB A, & direct> 1 1 1 1 1 1 1 1 1	A + val + C -> A ADDC A, <direct></direct>			
A - val -> A - v	ADDC A, WKI	2 1 0		
SUBB A, # All			0011nnn	
SUBB A, # All	A - val -> A SUBB A, <direct></direct>	2 1 1	0010101 <direct></direct>	
Increment	SUBB A, # <data></data>			
Increment	INC A	1 1 0	0000100	
INC 9R1	INC Rn	1 1 0		
NC DPTR	Increment INC <direct></direct>	2 1 0	0000101 <direct></direct>	
DEC Rn DEC Rn DEC Gliest> 1 1 00001001 cdreet> DEC Gliest> 2 1 00010101 cdreet> DEC Gliest> 2 1 0001001 cdreet> DEC Gliest> DEC G	INC DPTR	1 2 1	0100011	
DEC < colspan="2">DEC < colspan="2">DEC < colspan="2"> colspan="2">DEC < colspan="2"> colspan="2" colspan="2"> colspan="2" colspa	DEC A			
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	Decrement DEC Rn		0011nnn 0010101	
A/B -> A , A/B -> B	DEC @Ri	1 1 0	0010111	
MOV GRI, # < data>	Multiply: low -> A, high -> B MUL AB	1 4 1		
MOV A. #<-data>	A/B-> A, A%B-> B DIV AB			
MOV A. #<-data>	MOV @Ri, # <data></data>	1 1 1 1	111011i	
MOV DPTR, #: clata 3	MOV @Ri, <direct></direct>	2 2 1	010011i <src></src>	
MOV DPTR, #: clata 3	MOV A, # <data></data>	2 1 0	1110100 <data></data>	
MOV DPTR, #: clata 3	MOV A, with	2 1 1	1100111 1100101 <src></src>	
MOV DPTR, #: clata 3	MOV A, Rn	1 1 1	1101nnn	
MOV DPTR, #: clata 3	dest <- src MOV <direct>, <direct< td=""><td>t> 3 2 1</td><td></td><td><src></src></td></direct<></direct>	t> 3 2 1		<src></src>
MOV DPTR, #: clata 3	MOV <direct>, #<da MOV <direct> @Di</direct></da </direct>	a> 3 2 0	1110101 <dest></dest>	<data></data>
MOV DPTR, #: clata 3	MOV <direct>, A</direct>	2 1 1	1110101 <dest></dest>	1
MOV Ra. # Adata MOV Ra. # <a block"="" href="https://data.ni</td><td></td><td></td><td>0001nnn <dest></td><td></td></tr><tr><td><math display=">\begin{array}{c ccccccccccccccccccccccccccccccccccc	MOV DPTR, # <data></data>	2 1 0	Illinnn (data)	<data low></data low>
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	MOV Rn, A	1 i i	1111nnn	1
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	MOV Rn, <direct></direct>	2 2 1	0101nnn <src></src>	1
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	no fucking clue MOVC A, @A+DPTR MOVC A, @A+PC	1 2 1	0000011	1
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	MOVX @Ri, A		111001i	+
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	ACC <-> ext. memory MOVX A, @DPTR	1 2 1	1100000	1
Sease Operations				+
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	stack operations POP <direct></direct>		1010000 <dest></dest>	1
	XCH A, @Ri	1 1 1	100011i	1
	exchange A and src XCH A, <direct></direct>	2 1 1		1
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	exchange low nibble XCHD A GR	1 1 1	1010111	+
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	ACALL <direct(11b)></direct(11b)>	2 2 a	aa10001 <addr low=""></addr>	
RET	LCALL <direct(16b)></direct(16b)>	3 2 0	0010010 <addr high=""></addr>	<addr low></addr low>
$ \begin{array}{c cccc} AJMP \cdot Cdirect(11b) \rangle & 2 & 2 & aaa00001 & $				1
$ \begin{array}{c cccc} LIMP < direct(16b) > & 3 & 2 & 00000010 & < addr higher) \\ SIMP (< cfiest(8b) > & 2 & 2 & 100000010 & < field higher) \\ IMP (8A + DPTR) & 1 & 2 & 011100011 & < cfiest > 12 & 01110011 & < field higher) \\ IZ (< cfiest(8b) > & 2 & 2 & 011100001 & < cfiest > 12 & 01110011 & <$	AJMP < direct(11b)>	2 2 a	a000001 <addrllow></addrllow>	+
SJMP < offset(8b)	LIMP < direct(16b)>	3 2 0	0000010 <addr high=""></addr>	<addr low></addr low>
JZ <offset(8b)> 2 2 01100000 <offset></offset></offset(8b)>	SJMP <offset(8b)></offset(8b)>	2 2 1	0000000 <offset></offset>	1
	JZ <offset(8b)></offset(8b)>	2 2 0	1100000 <offset></offset>	1
INZ <offset(8h)> 2 2 01110000 <offset></offset></offset(8h)>	INZ <offeet(8h)></offeet(8h)>	2 2 0	1110000 <offset></offset>	1
jumping and branching CJNE sqlt, #cdata>, coffset(8b)> 3 2 10110100 cdata> CJNE A, #cdata>, coffset(8b) 3 2 1011011 cdata> CJNE A, #cdata>, coffset(8b) 3 2 1011010 cdata> CJNE A, clirecty, coffset(8b) 3 2 1011010 csrc>	jumping and branching CJNE @Ri, # <data>,</data>			<offset></offset>
CJNE A, # <data>, <offset(8b) 10110100="" 2="" 3="" <data=""> CJNE A, <direct>, <offset(8b) 10110101="" 2="" 3="" <src=""></offset(8b)></direct></offset(8b)></data>	CINE A, # <data>, <</data>	<offset(8b)> 3 2 1</offset(8b)>	011011i <data></data>	<offset></offset>
CJNE Rn, # <data>, <offset(8b) 10111nnn="" 2="" 3="" <data="" =""></offset(8b)></data>	CJNE Rn, # <data>.</data>	<offset(8b)> 3 2 1 ffset(8b) 3 2 1 ffset(8b) 2 2 1</offset(8b)>	0110100 <data></data>	
DJNZ < direct > < offset(8b) > 3 2 11010101 < src >	DJNZ <direct>, <offs< td=""><td>offset(8b) 3 2 1</td><td>0110100 <data> 0110101 <src></src></data></td><td><offset></offset></td></offs<></direct>	offset(8b) 3 2 1	0110100 <data> 0110101 <src></src></data>	<offset></offset>
DJNZ Rn, <offset(8b)> 2 2 11011nnn <offset> do nothing NOP 1 1 00000000 </offset></offset(8b)>	DJNZ Rn, <offset(8b)< td=""><td>offset(8b) 3 2 1 t(8b)> 3 2 1</td><td>0110100</td><td></td></offset(8b)<>	offset(8b) 3 2 1 t(8b)> 3 2 1	0110100	

II. 8051 Organization

	9
ROM	4K
RAM	128B
Address Bus	16b
I/O	4x 8b parallel ports, 1x serial port (all bit-addressable)

A. Components

$\begin{array}{c} \begin{tabular}{ll} \begin{tabular}{ll$

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

B. Program Flow

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

C. Addressing Modes

```
Implied Mode Immediate Mode Immediate Mode operands are specified implicitly operands evaluate to value operands evaluate to value operands evaluate to value operands evaluate to value operands of the index register and a displacement operand is a register containing wanted value 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