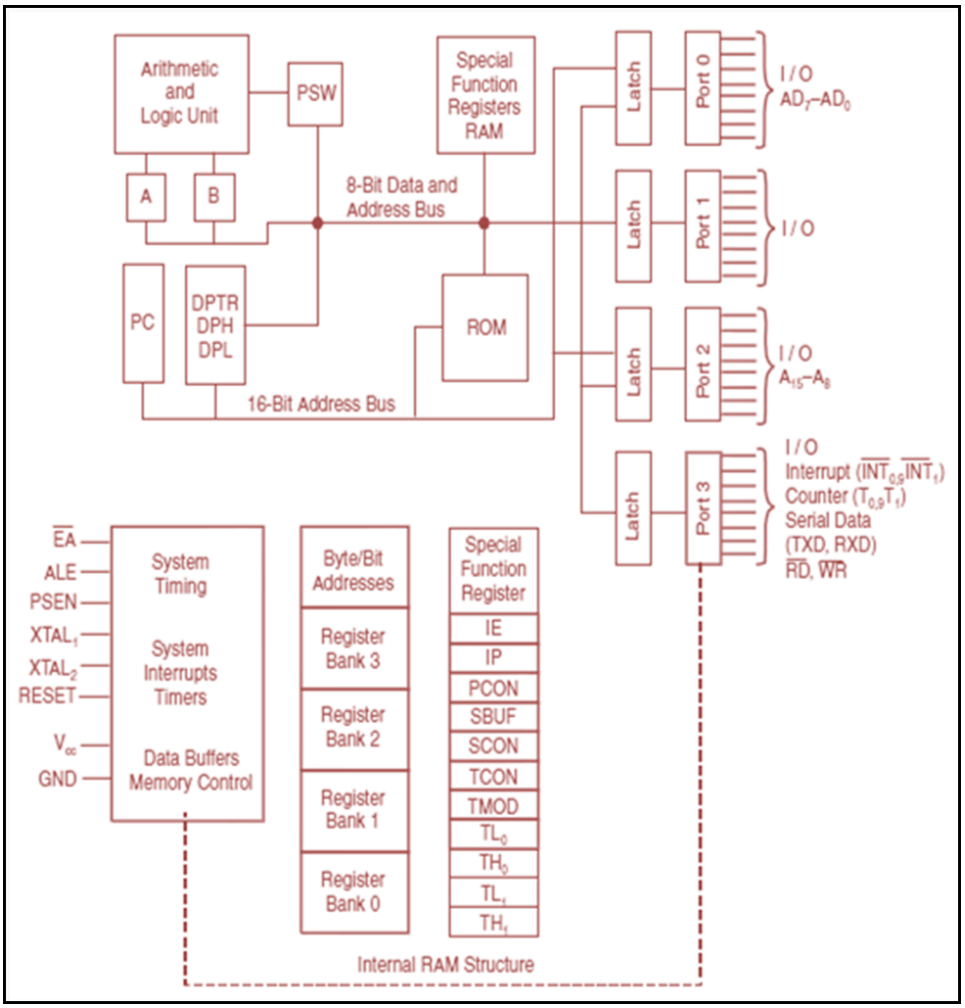

Fig. Architecture of 8051 microcontroller

Instruction Set of 8051

8051 INSTRUCTION SET

ARITHMETIC INSTRUCTIONS

Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication etc. After execution, the result is stored in the first operand. For example:

ADD A,R1 - The result of addition (A+R1) will be stored in the accumulator.

Arithmetic Instructions

Mnemonic	*Description*	*Byte*	*Cycle*
ADD A,Rn	Adds the register to the accumulator	1	1
ADD A,direct	Adds the direct byte to the accumulator	2	2
ADD A,@Ri	Adds the indirect RAM to the accumulator	1	2
ADD A,#data	Adds the immediate data to the accumulator	2	2
ADDC A,Rn	Adds the register to the accumulator with a carry flag	1	1
ADDC A,direct	Adds the direct byte to the accumulator with a carry flag	2	2
ADDC A,@Ri	Adds the indirect RAM to the accumulator with a carry flag	1	2
ADDC A,#data	Adds the immediate data to the accumulator with a carry flag	2	2
SUBB A,Rn	Subtracts the register from the accumulator with a borrow	1	1
SUBB A,direct	Subtracts the direct byte from the accumulator with a borrow	2	2
SUBB A,@Ri	Subtracts the indirect RAM from the accumulator with a borrow	1	2
SUBB A,#data	Subtracts the immediate data from the accumulator with a borrow	2	2
INC A	Increments the accumulator by 1	1	1
INC Rn	Increments the register by 1	1	2

CJNE @Ri,#data,rel	Compares immediate data to indirect register and jumps if not equal. Short jump.	3	
DJNZ Rn,rel	Decrements register and jumps if not 0. Short jump.	2	
DJNZ Rx,rel	Decrements direct byte and jump if not 0. Short jump.	3	
NOP	No operation	1	

Data Transfer Instructions

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix "X" (MOVX), the data is exchanged with external memory.

Data Transfer Instructions

Mnemonic	*Description*	*Byte*	*Cycle*
MOV A,Rn	Moves the register to the accumulator	1	
MOV A,direct	Moves the direct byte to the accumulator	2	
MOV A,@Ri	Moves the indirect RAM to the accumulator	1	
MOV A,#data	Moves the immediate data to the accumulator	2	
MOV Rn,A	Moves the accumulator to the register	1	
MOV Rn,direct	Moves the direct byte to the register	2	
MOV Rn,#data	Moves the immediate data to the register	2	
MOV direct,A	Moves the accumulator to the direct byte	2	
MOV direct,Rn	Moves the register to the direct byte	2	
MOV direct,direct	Moves the direct byte to the direct byte	3	
MOV direct,@Ri	Moves the indirect RAM to the direct byte	2	
MOV direct,#data	Moves the immediate data to the direct byte	3	
MOV @Ri,A	Moves the accumulator to the indirect RAM	1	
MOV @Ri,direct	Moves the direct byte to the indirect RAM	2	
MOV @Ri,#data	Moves the immediate data to the indirect RAM	2	
MOV DPTR,#data	Moves a 16-bit data to the data pointer	3	
MOVC A,@A+DPTR	Moves the code byte relative to the DPTR to the accumulator (address=A+DPTR)	1	
MOVC A,@A+PC	Moves the code byte relative to the PC to the accumulator (address=A+PC)	1	
MOVX A,@Ri	Moves the external RAM (8-bit address) to the accumulator	1	
MOVX A,@DPTR	Moves the external RAM (16-bit address) to the accumulator	1	
MOVX @Ri,A	Moves the accumulator to the external RAM (8-bit address)	1	
MOVX @DPTR,A	Moves the accumulator to the external RAM (16-bit address)	1	

INC Rx	Increments the direct byte by 1	2	3
INC @Ri	Increments the indirect RAM by 1	1	3
DEC A	Decrements the accumulator by 1	1	1
DEC Rn	Decrements the register by 1	1	1
DEC Rx	Decrements the direct byte by 1	2	2
DEC @Ri	Decrements the indirect RAM by 1	1	3
INC DPTR	Increments the Data Pointer by 1	1	3
MUL AB	Multiplies A and B	1	5
DIV AB	Divides A by B	1	5
DA A	Decimal adjustment of the accumulator according to BCD code	1	1

Branch Instructions

There are two kinds of branch instructions:

Unconditional jump instructions: upon their execution a jump to a new location from where the program continues execution is executed.

Conditional jump instructions: a jump to a new program location is executed only if a specified condition is met. Otherwise, the program normally proceeds with the next instruction.

Branch Instructions

Mnemonic	*Description*	*Byte*	*Cycle*
ACALL addr11	Absolute subroutine call	2	6
LCALL addr16	Long subroutine call	3	6
RET	Returns from subroutine	1	4
RETI	Returns from interrupt subroutine	1	4
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (from –128 to +127 locations relative to the following instruction)	2	3
JC rel	Jump if carry flag is set. Short jump.	2	3
JNC rel	Jump if carry flag is not set. Short jump.	2	3
JB bit,rel	Jump if direct bit is set. Short jump.	3	4
JBC bit,rel	Jump if direct bit is set and clears bit. Short jump.	3	4
JMP @A+DPTR	Jump indirect relative to the DPTR	1	2
JZ rel	Jump if the accumulator is zero. Short jump.	2	3
JNZ rel	Jump if the accumulator is not zero. Short jump.	2	3
CJNE A,direct,rel	Compares direct byte to the accumulator and jumps if not equal. Short jump.	3	4
CJNE A,#data,rel	Compares immediate data to the accumulator and jumps if not equal. Short jump.	3	4
CJNE Rn,#data,rel	Compares immediate data to the register and jumps		

PUSH direct	Pushes the direct byte onto the stack	2
POP direct	Pops the direct byte from the stack/td>	2
XCH A,Rn	Exchanges the register with the accumulator	1
XCH A,direct	Exchanges the direct byte with the accumulator	2
XCH A,@Ri	Exchanges the indirect RAM with the accumulator	1
XCHD A,@Ri	Exchanges the low-order nibble indirect RAM with the accumulator	1

Logic Instructions

Logic instructions perform logic operations upon corresponding bits of two registe
After execution, the result is stored in the first operand.

Logic Instructions

Mnemonic	*Description*	*Byte*	*Cyc*
ANL A,Rn	AND register to accumulator	1	
ANL A,direct	AND direct byte to accumulator	2	
ANL A,@Ri	AND indirect RAM to accumulator	1	
ANL A,#data	AND immediate data to accumulator	2	
ANL direct,A	AND accumulator to direct byte	2	
ANL direct,#data	AND immediate data to direct register	3	
ORL A,Rn	OR register to accumulator	1	
ORL A,direct	OR direct byte to accumulator	2	
ORL A,@Ri	OR indirect RAM to accumulator	1	
ORL direct,A	OR accumulator to direct byte	2	
ORL direct,#data	OR immediate data to direct byte	3	
XRL A,Rn	Exclusive OR register to accumulator	1	
XRL A,direct	Exclusive OR direct byte to accumulator	2	
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	1	
XRL A,#data	Exclusive OR immediate data to accumulator	2	
XRL direct,A	Exclusive OR accumulator to direct byte	2	
XORL direct,#data	Exclusive OR immediate data to direct byte	3	
CLR A	Clears the accumulator	1	
CPL A	Complements the accumulator (1=0, 0=1)	1	
SWAP A	Swaps nibbles within the accumulator	1	
RL A	Rotates bits in the accumulator left	1	
RLC A	Rotates bits in the accumulator left through carry	1	
RR A	Rotates bits in the accumulator right	1	
RRC A	Rotates bits in the accumulator right through carry	1	

Bit-oriented Instructions

Similar to logic instructions, bit-oriented instructions perform logic operations. The difference is that these are performed upon single bits.

Bit-oriented Instructions

Mnemonic	*Description*	*Byte*	*Cycle*
CLR C	Clears the carry flag	1	
CLR bit	Clears the direct bit	2	
SETB C	Sets the carry flag	1	
SETB bit	Sets the direct bit	2	
CPL C	Complements the carry flag	1	
CPL bit	Complements the direct bit	2	
ANL C,bit	AND direct bit to the carry flag	2	
ANL C,/bit	AND complements of direct bit to the carry flag	2	
ORL C,bit	OR direct bit to the carry flag	2	
ORL C,/bit	OR complements of direct bit to the carry flag	2	
MOV C,bit	Moves the direct bit to the carry flag	2	
MOV bit,C	Moves the carry flag to the direct bit	2	

APPENDIX-II

DESCRIPTION OF ALL 8051 INSTRUCTIONS

Here is a list of the operands and their meanings:

- **A** - accumulator;
  Rn - is one of working registers (R0-R7) in the currently active RAM memory bank;
- **Direct** - is any 8-bit address register of RAM. It can be any general-purpose register or a SFR (I/O port, control register etc.);
- **@Ri** - is indirect internal or external RAM location addressed by register R0 or R1;
- **#data** - is an 8-bit constant included in instruction (0-255);
- **#data16** - is a 16-bit constant included as bytes 2 and 3 in instruction (0-65535);
- **addr16** - is a 16-bit address. May be anywhere within 64KB of program memory;
- **addr11** - is an 11-bit address. May be within the same 2KB page of program memory as the first byte of the following instruction;