8051 Instruction Set Reference

NOP - No Operation

Description Literally does nothing but waste time for an entire machine cycle. Commonly used for delays.

Syntax
```
NOP
```

ANL - Bitwise AND

Description Performs a bitwise logical AND operation. Only the first operand is affected.

Syntax
```
ANL addr, A
ANL addr, #imm
ANL A, #imm
ANL A, addr
ANL A, @R0
ANL A, @R1
ANL A, R[0...7]
```

ORL - Bitwise OR

Description Performs a bitwise logical OR operation. Only the first operand is affected.

Syntax
```
ORL addr, A
ORL addr, #imm
ORL A, #imm
ORL A, addr
ORL A, @R0
ORL A, @R1
ORL A, R[0...7]
```

XRL - Bitwise Exclusive OR

Description Performs a bitwise logical XOR operation. Only the first operand is affected.

Syntax
```
XRL addr, A
XRL addr, #imm
XRL A, #imm
XRL A, addr
XRL A, @R0
XRL A, @R1
XRL A, R[0...7]
```

ADD, **ADDC** - Add Accumulator, Add Accumulator With Carry

Description Both ADD and ADDC add the value of the given operand to the accumulator. The value of the operand is not affected. They are pretty much the same except that ADDC also adds the value of the carry flag to the accumulator as well as the operand.

Syntax

```
ADD addr, A                 ADDC addr, A
ADD addr, #imm              ADDC addr, #imm
ADD A, #imm                 ADDC A, #imm
ADD A, addr                 ADDC A, addr
ADD A, @R0                  ADDC A, @R0
ADD A, @R1                  ADDC A, @R1
ADD A, R[0...7]             ADDC A, R[0...7]
```

SUBB - Subtract From Accumulator With Borrow

Description Subtracts the given value from the value in the accumulator.

Syntax

```
SUBB addr, A
SUBB addr, #imm
SUBB A, #imm
SUBB A, addr
SUBB A, @R0
SUBB A, @R1
SUBB A, R[0...7]
```

MUL - Multiply Accumulator by B

Description Multiplies the *unsigned* value in the accumulator (A) by the *unsigned* value in B. Note that multiplying two n-bit numbers together will produce a 2 × n-bit number. The lower-order byte is placed in the accumulator and the higher-order byte is placed in B. Both registers are affected.

Syntax

```
MUL AB
```

DIV - Divide Accumulator by B

Description Divides the *unsigned* value in the accumulator (A) by the *unsigned* value in B. Note that dividing two n-bit numbers will produce an n-bit quotient and an n-bit remainder. The quotient is placed in the accumulator and the remainder in B. Both registers are affected.

Syntax

```
DIV AB
```

INC - Increment Register

Description Increments the value in the given register. Note that this does not affect the carry flag if the result rolls over from the maximum value to 0.

Syntax

```
INC A
INC addr
INC @R0
INC @R1
INC R[0...7]
INC DPTR
```

DEC - Decrement Register

Description Decrements the value in the given register. Note that this does not affect the carry flag if the result rolls over from 0 to the maximum value.

Syntax
```
DEC A
DEC addr
DEC @R0
DEC @R1
DEC R[0...7]
```

RL - Rotate Accumulator Left

Description Shifts the bits of the accumulator to the left. The left-most bit is loaded into the right-most bit.

Syntax
```
RL A
```

RLC - Rotate Accumulator Left Through Carry

Description Shifts the bits of the accumulator to the left. The carry flag is loaded into the right-most bit and *then* the left-most bit is loaded into the carry flag. Useful for multiplying by 2.

Syntax
```
RL A
```

RR - Rotate Accumulator Right

Description Shifts the bits of the accumulator to the right. The right-most bit is loaded into the left-most bit.

Syntax
```
RR A
```

RRC - Rotate Accumulator Right Through Carry

Description Shifts the bits of the accumulator to the right. The carry flag is loaded into the left-most bit and *then* the right-most bit is loaded into the carry flag. Useful for dividing by 2.

Syntax
```
RL A
```

AJMP - Absolute Jump

Description Absolutely, unconditionally jumps to the given address by setting the program counter to it and changing the page.

Syntax
```
AJMP addr
```

SJMP - Short Jump

Description Absolutely, unconditionally jumps to the given address by setting the program counter to it. The address must be within -128 to +128 bytes of the following instruction.

Syntax
```
SJMP addr
```

LJMP - Long Jump

Description Absolutely, unconditionally jumps to the given address by setting the program counter to it.

Syntax
```
LJMP addr
```

JMP - Jump to Address

Description Absolutely, unconditionally jumps to the address computed as the sum of the DPTR and the value of the Accumulator.

Syntax
```
LJMP @A+DPTR
```

CJNE - Compare and Jump if Not Equal

Description Compares the value of the first two operands and jumps to the given address if they are NOT equal.

Syntax
```
CJNE A, #imm, addr
CJNE A, addr, addr
CJNE @R0, #imm, addr
CJNE @R1, #imm, addr
CJNE R[0...7], #imm, addr
```

DJNZ - Decrement Register and Jump if Not Zero

Description Decrements the value of the register. If the new value is not zero, then jump to the given address.

Syntax
```
DJNZ addr, addr
DJNZ R[0...7], addr
```

JB - Jump if Bit Set

Description If the bit at the given bit address is set, then jump to the given address.

Syntax
```
JB bit-addr, addr
```

JNB - Jump if Bit Not Set

Description If the bit at the given bit address is NOT set, then jump to the given address.

Syntax
```
JNB bit-addr, addr
```

JBC - Jump if Bit Set and Clear Bit

Description If the bit at the given bit address is set, clear the bit and then jump to the given address.

Syntax
```
JBC bit-addr, addr
```

JC - Jump if Carry Set

Description If the carry bit is set, then jump to the given address.

Syntax
```
JC addr
```

JNC - Jump if Carry Not Set

Description If the carry bit is NOT set, then jump to the given address.

Syntax
```
JNC addr
```

JZ - Jump if Accumulator Zero

Description If the value in the accumulator is zero, then jump to the given address.

Syntax
```
JZ addr
```

JNZ - Jump if Accumulator Not Zero

Description If the value in the accumulator is NOT zero, then jump to the given address.

Syntax `JNZ addr`

ACALL - Absolute Call

Description Absolutely, unconditionally calls a subroutine at the given code address. This happens in two steps:

1. The address of the next instruction (directly after ACALL) is pushed onto the stack.
2. The program counter is changed to the given code address.

Uses only 2 bytes and can only specify addresses in the 2K-byte range around it.

Syntax `ACALL addr`

LCALL - Long Call

Description Absolutely, unconditionally calls a subroutine at the given code address. This happens in two steps:

1. The address of the next instruction (directly after ACALL) is pushed onto the stack.
2. The program counter is changed to the given code address.

Uses 3 bytes and can specify addresses anywhere in the 8051's 64K-byte internal memory.

Syntax `LCALL addr`

RET - Return From Subroutine

Description Returns from a subroutine called by ACALL or LCALL. This happens in two steps:

1. An address is popped off the stack.
2. The program counter is changed to that address.

Syntax `RET`

RETI - Return From Interrupt

Description Returns from an interrupt service routine. This happens in three steps:

1. Interrupts of equal or lower priorities to the current interrupt that is terminating are enabled.
2. An address is popped off the stack.
3. The program counter is changed to that address.

Syntax `RETI`

PUSH - Push Value Onto Stack

Descriptor Pushes the value at the given address onto the stack. This happens in two steps:

1. The stack pointer is incremented.
2. The value at the given address is copied to the stack pointer's address.

Syntax `PUSH addr`

POP - Pop Value From Stack

Descriptor Pops a value from the stack into the given address. This happens in two steps:

1. The value at the stack pointer's address is copied to the given address.
2. The stack pointer is decremented.

Syntax
```
POP addr
```

MOV - Move Memory

Descriptor Copies the value from the second operand into the first.

Syntax
```
MOV @R0, #imm
MOV @R1, #imm
MOV @R0, A
MOV @R1, A
MOV @R0, addr
MOV @R1, addr
MOV A, #imm
MOV A, @R0
MOV A, @R1
MOV A, R[0...7]
MOV A, addr
MOV C, bit-addr
MOV DPTR, #imm
MOV R[0...7], #imm
MOV R[0...7], A
MOV R[0...7], addr
MOV bit-addr, C
MOV addr, #imm
MOV addr, @R0
MOV addr, @R1
MOV addr, R[0...7]
MOV addr, A
MOV addr, addr
```

MOVC - Move Code Memory

Descriptor Copies a value from code memory into the accumulator.

Syntax
```
MOVC A, @A+DPTR
MOVC A, @A+PC
```

MOVX - Move Extended Memory

Descriptor Moves a byte to/from external memory from/to the accumulator.

Syntax
```
MOVX @DPTR, A
MOVX @R0, A
MOVX @R1, A
MOVX A, @DPTR
MOVX A, @R0
MOVX A, @R1
```

SETB - Set Bit

Descriptor Sets the given bit.

Syntax
```
SETB C
SETB bit-addr
```

CLR - Clear Register

Descriptor For an individual bit, sets its value to 0. For a register, sets all of its bits to 0.

Syntax
```
CLR bit-addr
CLR C
CLR A
```

CPL - Complement Register

Descriptor For an individual bit, complements (flips) its value. For a register, complements (flips) all of its bits.

Syntax
```
CPL bit-addr
CPL C
CPL A
```

DA - Decimal Adjust

Descriptor Adjusts the contents of the Accumulator to correspond to a BCD (Binary Coded Decimal) number after two BCD numbers have been added by the ADD or ADDC instruction. If the carry bit is set or if the value of bits 0-3 exceed 9, 0x06 is added to the accumulator. If the carry bit was set when the instruction began, or if 0x06 was added to the accumulator in the first step, 0x60 is added to the accumulator.

Syntax
```
CPL bit-addr
CPL C
CPL A
```

SWAP - Swap Accumulator Nibbles

Descriptor In the accumulator, swaps bits 0-3 with 4-7.

Syntax
```
SWAP A
```

XCH - Exchange Bytes

Descriptor Swaps the contents of the acccumulator with another register. This happens simultaneously and therefore does not require a third register to do the swap.

Syntax
```
XCH A, @R0
XCH A, @R1
XCH A, R[0...7]
XCH A, addr
```

XCHD - Exchange Digits

Descriptor Swaps bits 0-3 of the accumulator with bits 0-3 of an internal memory address. Bits 4-7 are unaffected.

Syntax
```
XCHD A, @R0
XCHD A, @R1
```

Undefined - Undefined Instruction

Descriptor This instruction is undocumented. The 8051 has 256 instructions and opcode 0xA5 is the only one that is not used. Try not to use it.

Syntax ???