**Instruction Set**

* **Compare and Jump if Not Equal CNJE A,#K , DIR**
  + *CJNE compares the magnitudes of the first two operands and branches if their values are not equal.*
    - *B <- K*
    - *IF A == B THEN*
      * *PC <- PC +1*
    - *ELSE*
      * *PC <- DIR*
* ***CALL CALL DIR***
  + *Call calls a subroutine located at the indicated address. The instruction adds one to the program counter to generate the address of the next instruction and then pushes the address onto the stack, decrementing the stack pointer. The address provided by the instruction its fetch onto the program counter to continue with the subroutine.*
    - *PC <- PC + 1*
    - *RAM(SPTR) <- PC*
    - *SPTR <- SPTR - 1*
    - *PC <- DIR*
* ***RETURN*** 
  + *Return pops the address of the PC successively from the stack, incrementing the SPTR by one. Program execution continues at the resulting address, generally the instruction it’s used after a CALL.*
    - *STPR <- STPR + 1*
    - *PC <- RAM(SPTR)*
* ***SWAP*** 
  + *Swap interchanges nibbles within the accumulator, register A. This operation can be considered as a 4 bit rotate instruction.*
* **ROTATE LEFT ROTL**
  + The eight bits in the accumulator are rotated one bit to the left. Bit 7 is rotate to the carry (C), and C it’s rotate to Bit 0.
    - C <-
    - <-
    - <- C
* **ROTATE RIGHT ROTR**
  + The eight bits in the accumulator are rotated one bit to the right. C it’s rotate to Bit 7 and Bit 0 is rotate the carry.
    - C <-
    - <-
    - <- C
* ***JUMP JMP DIR***
  + *Program control branches unconditionally to the address indicated. The PC is fetched with a 7 bit address provided by the instruction.*
    - *PC <- DIR*
* ***MOVE MOV <src> , <dest>***
  + The byte variable indicated by the first operand is copied into the location specified by the second operand. The source is not affected.   
    This is a very flexible operation, there is up to six variations of this instruction, allowing different combination of sources and destinations.
    - <src> <- <dest>
      * MOV A,B
      * MOV A,DIR
      * MOV A ,Pn
      * MOV B,A
      * MOV DIR,A
      * MOV PN,A
* **ADD SUM** 
  + ADD adds the auxiliary register B to the Accumulator, leaving the result in the accumulator. The carry and auxiliary carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise.
    - A<- A + B
* **SUBTRACT SUB**
  + SUB subtracts the auxiliary register B to the Accumulator, leaving the result in the accumulator. The carry flag is set if there is a borrow from bit 7 and cleared otherwise.
    - A<- A – B
* **PRODUCT MULT**
  + MULT performs the integer product between the low part of the accumulator and the auxiliary register B, leaving the result in the accumulator. No flag is affected by this operation.
    - A <-
* **DIVISION DIV**
  + DIV performs the integer division between the low part of the accumulator and the auxiliary register B, leaving the result in the accumulator. No flag is affected by this operation.
    - A<- /
* **NO OPERATION NOP**
  + Execution continues at the following instruction. Other than the PC, no registers or flags are affected.
    - PC <- PC + 1
* **AND AND A,K**
  + AND performs a bit-wise and operation between the accumulator and an immediate data K, which is stored first in the auxiliary register B before the operation. No flag is affected by the instruction.
    - B <- K
    - A <- A & B
* **OR OR A,K**
  + OR performs a bit-wise or operation between the accumulator and an immediate data K, which is stored first in the auxiliary register B before the operation. No flag is affected by the instruction.
    - B <-K
    - A <- A | B
* **NOT NOT <src>**
  + NOT performs a bit-wise not operation with the source’s bits. The operand for this operation can be the accumulator or an immediate data K, which is stored first in the accumulator before the operation.
    - /A
* **XOR XOR A,K**
  + XOR performs a bit-wise xor operation between the accumulator and the immediate data K, which is stored first in the auxiliary register B before the operation. Equally flag is set when both operands are equal, and it’s cleared otherwise.
    - B <- K
    - A <

**Program Memory Structure**

|  |  |  |  |
| --- | --- | --- | --- |
| **DIR {6-4}** | **OC {5-0}** | **DIR {3-0}** | **DATA {3-0}** |
| **Address High Part** | **Operation Code** | **Address Low Part** | **Operand’s Data** |

16 Bits