**Practical – 4**

**AIM: Write an assembly language code in GNUsim8085 to implement Arithmetic instruction.**

|  |
| --- |
| **ADD**  **The content of operand are added to the content of the accumulator and**  **the result is stored in Accumulator.** |
| * **ADD R** is a mnemonic that stands for “Add contents of R to Accumulator”. * As addition is a binary operation, so it requires two operands to be operated on. * So input operands will reside on Accumulator and R registers and after addition the result will be stored back on to Accumulator. * In this case, “R” stands for any of the following registers or memory location M pointed by HL pair.   R = A, B, C, D, E, H, L, or M   * It is 1-Byte instruction so occupies only 1-Byte in memory. As R can have any of the eight values, there are eight opcodes for this type of instruction.  |  |  | | --- | --- | | **Mnemonics, Operand** | **Bytes** | | ADD A | 1 | | ADD B | 1 | | ADD C | 1 | | ADD D | 1 | | ADD E | 1 | | ADD H | 1 | | ADD L | 1 | | ADD M | 1 |  |  |  |  |  | | --- | --- | --- | --- | | ADD | R | A = A + R | ADD B | | ADD | M | A = A + Mc | ADD 2050 | |  |  |  |  | |
| **ADC**  **Addition with Carry** |
| * ADC is a mnemonic that stands for ‘ADd with Carry’ and ‘R’ stands for any of the following registers, or memory location M pointed by HL pair. * This instruction is mainly used to add contents of R register and Accumulator along with the carry value. * Here the carry can be either 0 or 1. * The result of this addition operation will be stored in the Accumulator itself overwriting its previous content. * This is 1-Byte instruction so also occupies 1-Byte in the memory. So this R can have 8 possible values and thus 8 possible opcodes.   R = A, B, C, D, E, H, L, or M   |  |  |  |  | | --- | --- | --- | --- | | ADC | R | A = A + R + prev. carry | ADC B | | ADC | M | A = A + Mc + prev. carry | ADC 2050 | |
| **ADI**  Add immediate means add an immediate value with the content of accumulator and it is stored in accumulator. |
| * **ADI** is a mnemonic, which stands for “ADd Immediate to Accumulator” and here “d8” stands for any 8-bit or 1-Byte of data. * This instruction is used to add 8-bit immediate data to the Accumulator. * The result of addition will be stored in the Accumulator. So the previous content of the Accumulator will be over written. * It occupies 2-Bytes in memory. The flags are affected based on the result.  |  |  |  |  | | --- | --- | --- | --- | | ADI | 8-bit data | A = A + 8-bit data | ADD 50 | |
| **ACI**  Add immediate to Accumulator with Carry |
| * **ACI** is a mnemonic which stands for 'Add with Carry Immediate to Accumulator' and here “d8” stands for any 8-bit or 1-Bytedata. * This instruction is actually meant for adding one 8-bit immediate data or operand to the Accumulator along with the carry value. * The result of the addition will be stored in the Accumulator itself and replacing initial value of the Accumulator. * As it is an arithmetic instruction, so the flags are affected based on the result. It holds 2-Bytes in the memory.  |  |  |  |  | | --- | --- | --- | --- | | ACI | 8-bit data | A = A + 8-bit data + prev. carry | ACI 50 | |
| **SUB**  **Subtract the content of a register or a memory location form the content of accumulator and the result is stored in the accumulator.** |
| * **SUB** is a mnemonic that stands for ‘SUBtract contents of R from Accumulator. Here R stands for any of the following registers, or memory location M pointed by HL pair.   R = A, B, C, D, E, H, L, or M   * SUB B (It subtracts the content of B register from the content of the accumulator.) * SUB M (It subtracts the content of memory location pointed by HL pair from the content ofaccumulator)  |  |  |  |  | | --- | --- | --- | --- | | SUB | R | A = A – R | SUB B | | SUB | M | A = A – Mc | SUB 2050 |  * 1 byte instruction * In this instruction content of the register R will get subtracted from the Accumulator and the resultant difference will be stored on the Accumulator replacing the previous content of the Accumulator. * Where R can be any of the eight values, so as a result there are eight opcode for this type of instruction as mentioned in the above table. It occupies only 1-Byte in the memory. |
| **SBB**  **Subtract with borrow** |
| SBB is a mnemonic that stands for 'SuBtract with Borrow' and 'R' stands for any of the following 7 registers, and also memory location M as pointed by HL register pair.  R = A, B, C, D, E, H, L, or M  This instruction is used to compute subtraction between contents of R register from the Accumulator's content, along with the carry (borrow) value.  The result of the subtraction will be stored in the Accumulator replacing Accumulator’s initial value.   |  |  |  |  | | --- | --- | --- | --- | | SBB | R | A = A – R – prev. carry | SBB B | | SBB | M | A = A – Mc -prev. carry | SBB 2050 | |
| **SUI** |
| * **SUI** is a mnemonic that stands for ‘SUbtract Immediate from Accumulator and * here **d8** stands for any 8-bit or 1-Byte data. * This instruction is used to subtract 8-bit immediate data from the Accumulator. * The result of the subtraction will be stored in the Accumulator over witting its previous content. As it is an arithmetic instruction, so flag bits are affected based on the result. * It is a 2-Byte instruction and occupies 2-Bytes in memory.  |  |  |  |  | | --- | --- | --- | --- | | SUI | 8-bit data | A = A – 8-bit data | SUI 50 |  * Example:   **MVI A, 50H**  **SUI 20H** |
| **SBI** |
| * **SBI** is a mnemonic that stands for “Subtract with Borrow Immediate from Accumulator” and here **d8** stands for any 8-bit data as operand. * This instruction is used to subtract 8-bit immediate data from the Accumulator along with the carry (borrow) value. * The result of subtraction will be stored in the Accumulator. As this is an arithmetic instruction, so the flags are affected based on the result produced. * It occupies 2 consecutive Bytes in memory.  |  |  |  |  | | --- | --- | --- | --- | | SBI | 8-bit data | A = A – 8-bit data – prev. carry | SBI 50 | |
| **DAD** |
| **DAD** is a mnemonic, which stands for **Double ADd** and also **rp** stands for any one of the following register pairs as mentioned below.  rp = BC, DE, or HL  As rp can have any of the three values, there are three opcodes for this type of instruction.  It occupies only 1-Byte in memory.  In this instruction HL register pair works as Accumulator. Because the 16-bit content of rp will be added with HL register pair content and sum thus produced will be stored back on to HL again. |
| **DAA** |
| **Example:**  **MVI A, 38H**  **MVI B, 45H**  **ADD B**  **DAA** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **Operand** | **Meaning** | **Explanation** |
| ADD | R  M | Add register or memory, to the accumulator | The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator.  **Example** − ADD K. |
| ADC | R  M | Add register to the accumulator with carry | The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator.  **Example** − ADC K |
| ADI | 8-bit data | Add the immediate to the accumulator | The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator.  **Example** − ADI 55K |
| ACI | 8-bit data | Add the immediate to the accumulator with carry | The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator.  **Example** − ACI 55K |
| DAD | Reg. pair | Add the register pair to H and L registers | The 16-bit data of the specified register pair are added to the contents of the HL register.  **Example** − DAD K |
| SUB | R  M | Subtract the register or the memory from the accumulator | The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator.  **Example** − SUB K |
| SBB | R  M | Subtract the source and borrow from the accumulator | The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator.  **Example** − SBB K |
| SUI | 8-bit data | Subtract the immediate from the accumulator | The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator.  **Example** − SUI 55K |
| SBI | 8-bit data | Subtract the immediate from the accumulator with borrow | The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.  **Example** − XCHG |
| INR | R  M | Increment the register or the memory by 1 | The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place.  **Example** − INR K |
| INX | R | Increment register pair by 1 | The contents of the designated register pair are incremented by 1 and their result is stored at the same place.  **Example** − INX K |
| DCR | R  M | Decrement the register or the memory by 1 | The contents of the designated register or memory are decremented by 1 and their result is stored at the same place.  **Example** − DCR K |
| DCX | R | Decrement the register pair by 1 | The contents of the designated register pair are decremented by 1 and their result is stored at the same place.  **Example** − DCX K |
| DAA | None | Decimal adjust accumulator | The contents of the accumulator are changed from a binary value to two 4-bit BCD digits.  If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits.  If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.  **Example** − DAA |

Add two 8-bit numbers

Statement: Add the contents of memory locations 4000H and 4001H and place the result in memory location 4002H.

1. Sample problem
2. (4000H) = 14H
3. (4001H) = 89H
4. Result = 14H + 89H = 9DH
6. Source program
7. LXI H 4000H : "HL points 4000H"
8. MOV A, M : "Get first operand"
9. INX H : "HL points 4001H"
10. ADD M : "Add second operand"
11. INX H : "HL points 4002H"
12. MOV M, A : "Store result at 4002H"
13. HLT : "Terminate program execution"

Subtract two 8-bit numbers

Statement: Subtract the contents of memory location 4001H from the memory location 2000H and place the result in memory location 4002H.

1. Program –: Subtract two 8-bit numbers
2. Sample problem:
3. (4000H) = 51H
4. (4001H) = 19H
5. Result = 51H – 19H = 38H
7. Source program:
8. LXI H, 4000H : "HL points 4000H"
9. MOV A, M : "Get first operand"
10. INX H : "HL points 4001H"
11. SUB M : "Subtract second operand"
12. INX H : "HL points 4002H"
13. MOV M, A : "Store result at 4002H"
14. HLT : "Terminate program execution