Experiment No: 5

**Date:**

**Write following 8085 assembly language programs with output.**

1. To add two arrays of five 8-bit values and store result in third array.
2. To store sum of unpacked BCD numbers from packed BCD numbers.
3. To store an ASCII value of each Hex digits.
4. To set an MSB of an 8-bit values if it has even no. of 1’s.
5. To reverse an array of ten 8-bit values using subroutine.
6. Find factorial of given 8-bit value.
7. To perform division of 16-bit number by 8-bit number.

**Competency and Practical Skills*:*** understanding of working of registers, instruction set of 8085

**Relevant CO:** CO2

**Objectives:**

1. To understand working of the 8085 instruction set.
2. To understand the behavior of the flags and how to use flag instructions.

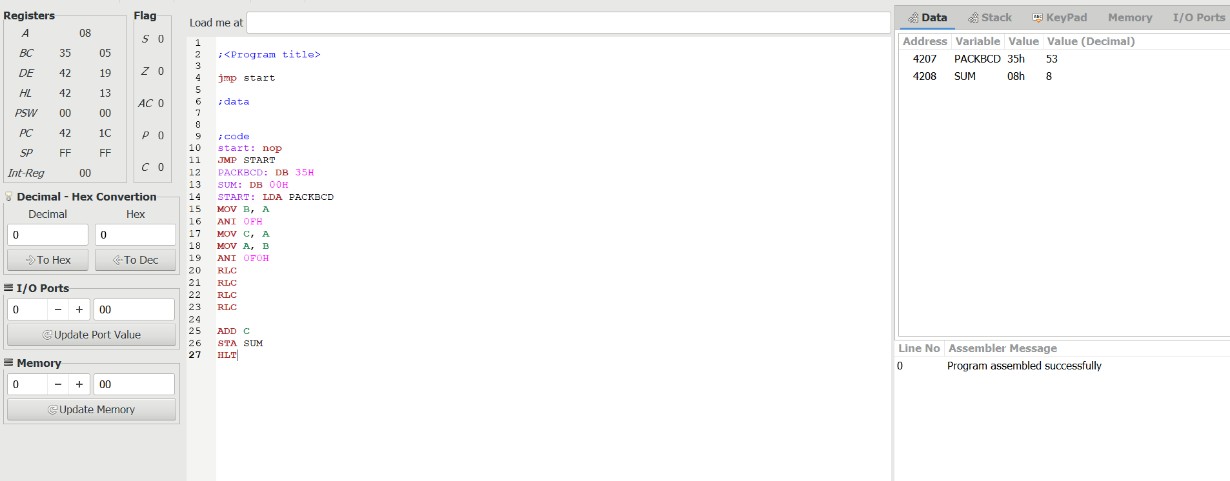
**Equipment/Instruments:** Personal Computer, GNUsim8085

**Explanation:**

1. **To add two arrays of five 8-bit values and store result in third array.**

|  |  |
| --- | --- |
| JMP START | *; Jump on start* |
| DATA1: DB 02, 04, 06, 08, 10, 07 | *; First Array* |
| DATA2: DB 01, 03, 05, 07, 09, 08 | *; Second Array* |
| SUM: DS 6 | *; Third array to store sum* |
| COUNT: DB 06 | *; Counter value* |
| START: LXI B, DATA1 | *; Load address of DATA1 in BC reg. pair* |
| LXI H, DATA2 | *; Load address of DATA2 in HL reg. pair* |
| LXI D, SUM | *; Load address of SUM in DE reg. pair* |
| LOOP: LDAX B | *; Start of LOOP; Read values from DATA1* |
| ADD M | *; Add values from DATA2* |
| STAX D | *; Store sum in SUM* |
| INX B | *; Increment location of DATA1* |
| INX H | *; Increment location of DATA2* |
| INX D | *; Increment location of SUM* |
| LDA COUNT | *; Load count in A register* |
| DCR A | *; Decrement count value in A register* |
| STA COUNT | *; Store count from A register* |
| JNZ LOOP | *; Check Zero flag, if it is not set go back to LOOP* |
| HLT | *; Stop the execution* |

**Output:**



1. **To store sum of unpacked BCD numbers from packed BCD numbers.**

*; Jump on start*

*; Packed BCD number*

*; To store sum of digits of Packed BCD number*

*; Load Packed BCD number to A register*

*; Copy number to B register*

*; Perform AND operation with 0Fh to clear first four bits*

*; Move last hex digit from A to C register*

*; Move Packed BCD number to A register*

*; Perform AND operation with 0F0h to clear last four bits*

*; Rotate A register value 1-bit left*

*; Rotate A register value 1-bit left*

*; Rotate A register value 1-bit left*

*; Rotate A register value 1-bit left (First hex digit is in last position)*

*; Add another digit from C register*

*; Store the sum on SUM*

*; Stop the execution*

JMP START PACKBCD: DB 35H SUM: DB 00H

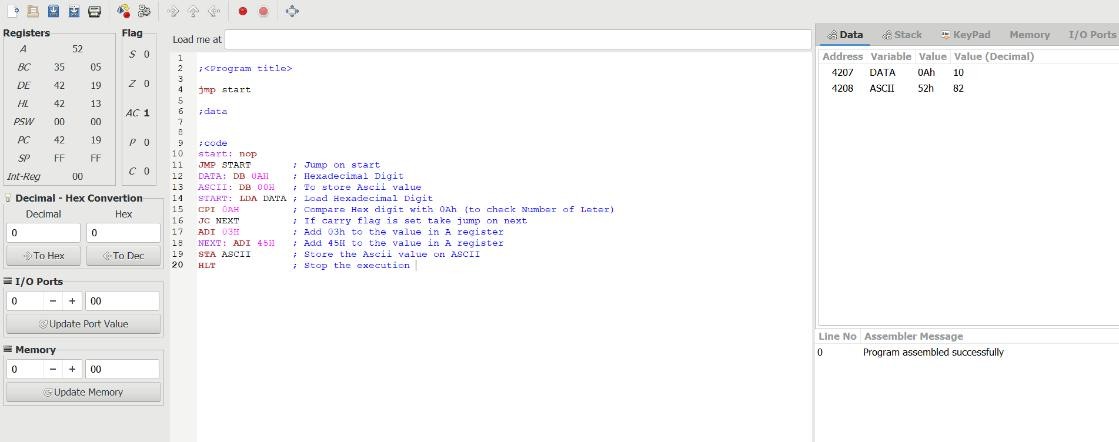
START: LDA PACKBCD MOV B, A

ANI 0FH MOV C, A MOV A, B ANI 0F0H RLC

RLC RLC RLC

ADD C STA SUM HLT

**Output:**



1. ***To store an ASCII value of each Hex digits.***

|  |  |
| --- | --- |
| JMP START | *; Jump on start* |
| DATA: DB 0AH | *; Hexadecimal Digit* |
| ASCII: DB 00H | *; To store Ascii value* |
| START: LDA DATA | *; Load Hexadecimal Digit* |
| CPI 0AH | *; Compare Hex digit with 0Ah (to check Number of Leter)* |
| JC NEXT | *; If carry flag is set take jump on next* |
| ADI 03H | *; Add 03h to the value in A register* |
| NEXT: ADI 45H | *; Add 45H to the value in A register* |
| STA ASCII | *; Store the Ascii value on ASCII* |
| HLT | *; Stop the execution* |

**Output:**

1. **To set an MSB of an 8-bit values if it has even no. of 1’s.**

*; Jump on start*

*; Store 8-bit number in NUM*

*; To store result*

*; Load NUM in A register*

*; Copy NUM from A register to D*

*; Clear B register value to store count of 1’s*

*; Store 09 in C register to rotate value 9 times right*

*; Rotate A register value right*

*; If carry flag is not set, take a jump on NEXT*

*; If carry flag is set, increment value of B register*

*; Decrement counter value in C register*

*; If Zero flag is not set, take a jump on LOOP*

*; Move count of 1’s to A register*

*; Rotate right to check LSB of count of 1’s*

*; if Carry flag is set, take jump on NEXT1 (no. of 1’s are odd)*

*; Restore NUM in A register*

*; Perform OR operation with 90H(1011010) to set MSB in NUM*

*; Store the result*

*; Stop the execution*

JMP START NUM: DB 10H RESULT: DB 00H

START: LDA NUM MOV D, A

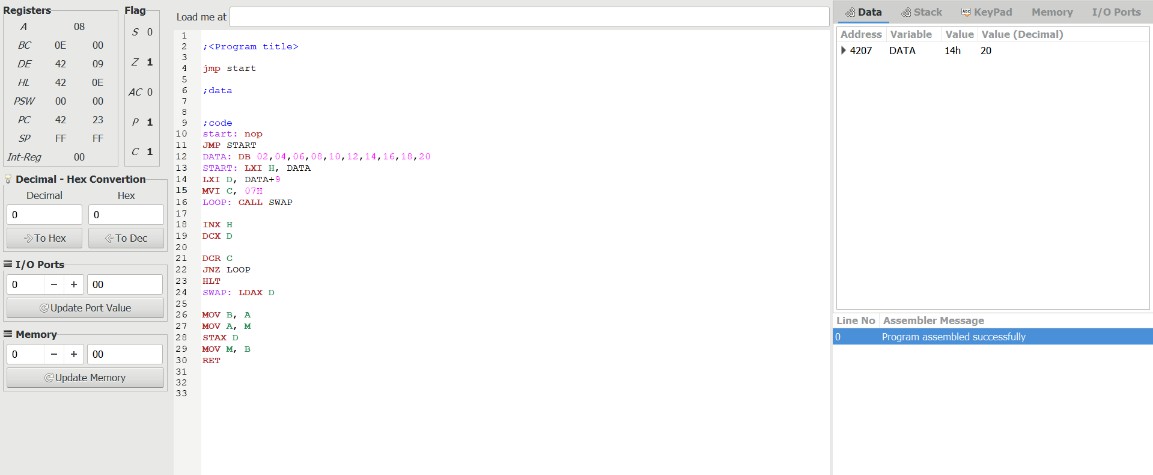
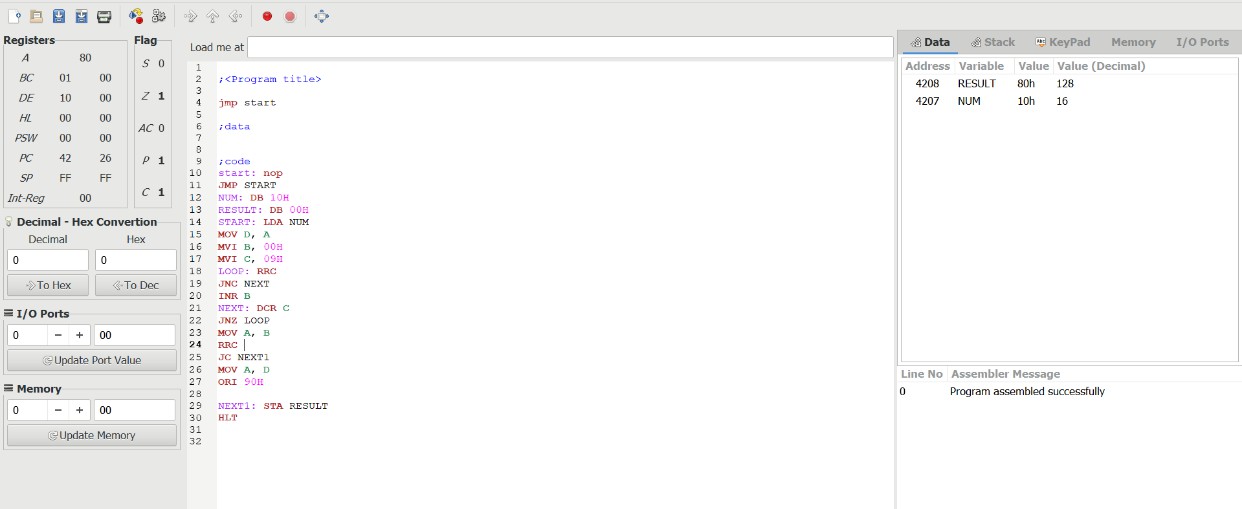
MVI B, 00H MVI C, 09H LOOP: RRC JNC NEXT INR B

NEXT: DCR C JNZ LOOP MOV A, B RRC

JC NEXT1 MOV A, D ORI 90H

NEXT1: STA RESULT

HLT



**Output:**

1. **To reverse an array of ten 8-bit values using subroutine*.***

*; Jump on start*

*; Store ten 8-bit values with DATA*

*; Load first address of DATA in HL reg pair*

*; Load last address of DATA in DE reg pair*

*; Set C register value to 07 for counter*

*; Call fun to swap values of address in HL and DE reg. pair*

*; Go to next location of DATA in forward*

*; Decrement address for location of DATA in*

*backward*

*; Decrement counter*

*; if Zero flag is not set, take a jump on LOOP*

*; Stop the execution*

*; Subroutine Load values from the location in DE reg. pair*

*; Move value to B register*

*; Move value to A from the location in HL reg. pair*

*; Store the value on the location in DE reg. pair*

*; Store the value on the location in HL reg. pair*

*; Return from the subroutine*

JMP START

DATA: DB 02, 04, 06, 08, 10, 12, 14,

16, 18, 20

START: LXI H, DATA LXI D, DATA+9

MVI C, 07H

LOOP: CALL SWAP

INX H

DCX D

DCR C JNZ LOOP HLT

SWAP: LDAX D

MOV B, A MOV A, M STAX D MOV M, B RET

**Output:**



1. ***Find factorial of given 8-bit value.***

|  |  |
| --- | --- |
| JMP START | *; Jump on start* |
| NUM: DB 09H | *; Store 8-bit number in NUM* |
| ANS: DB 00H | *; To store the factorial* |
| START: LDA NUM | *; Load NUM to A register* |
| MOV B, A | *; Move value to B from A reg.* |
| MVI D, 03H | *; Set value of D reg. to 03* |
| FACT: CALL MUL | *; Call subroutine* |
| DCR B | *; Decrement B register value* |
| JNZ FACT | *; If Zero flag is not jump on FACT* |
| MOV A, D | *; Copy value of D reg. to A* |
| STA ANS | *; Store the result in ANS* |
| HLT | *; Stop the execution* |
| MUL: MOV E, B | *; Subroutine Move value of B reg to A* |
| XRA A | *; Clear value of A reg* |
| ML: ADD D | *; Add value of D reg* |
| DCR E | *; Decrement value of E reg.* |
| JNZ ML | *; If Zero flag is not set, take jump on ML* |
| MOV D, A | *; Store sum in D reg.* |
| RET | *; Return from Subroutine* |

**Output:**

1. **To perform division of 16-bit number by 8-bit number*.***

|  |  |
| --- | --- |
| LXI H, 8000H | *; Point 8000H address* |
| MOV A, M | *; Store the lower order byte* |
| INX H | *; Increase the HL pair to point next loc* |
| MOV B, M | *; Store the higher order byte* |
| INX H | *; Increase the HL pair to point next loc* |
| MOV C, M | *; Load the denominator* |
| INR B | *; Increase B register* |
| LXI H, 0000H | *; Store 0000Hinto HL pair* |
| LOOP: SUB C | *; Subtract C from acc* |
| JC SKIP | *; Jump to SKIP when CY = 1* |
| INCR: INX H | *; Increase quotient part* |
| JMP LOOP | *; Jump to LOOP* |
| SKIP: DCR B | *; Decrease B* |
| JZ STORE | *; Jump to STORE when Z = 1* |
| JMP INCR | *; Jump to INCR* |
| STORE: ADD C | *; Add C with Acc* |
| XCHG | *; swap DE and HL pair contents* |
| LXI H, 8050H | *; Load the destination address* |
| MOV M, E | *; Store the lower order quotient* |
| INX H | *; Increase HL pair* |
| MOV M, D | *; Store the higher order quotient* |
| INX H | *; Increase HL pair* |
| MOV M, A | *; Store the remainder* |
| HLT | *; Stop the execution* |

**Output:**

**Conclusion:**

**Quiz:**

1. Explain XCHG instruction.
2. What is the purpose of the DAA instruction?
3. Explain JNC and JNZ instruction.

***Suggested Reference:***

* 1. R.S.Gaonkar, “Microprocessor Architecture, Programming and Applications with 8085A”, Penram International

***Rubric wise marks obtained:***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Knowledge of subject (2)** | | **Programming Skill** | | **Team work (2)** | | **Communication Skill (2)** | | **Ethics (2)** | |
| Good (2) | Average (1) | Good (2) | Average (1) | Good (2) | Satisfactory (1) | Good (2) | Satisfactory (1) | Good (2) | Average (1) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rubrics** | **1** | **2** | **3** | **4** | **5** | **Total** |
| **Marks** |  |  |  |  |  |  |