**Name of Student : Jay Makwana**

**Class and Division: CySe, div: 15**

**Roll Number: 20**

**Date : 03/02/2022**

**Time: 11:15am to 12:15pm**

##### **MAHAVIR EDUCATION TRUST’S**

##### **SHAH AND ANCHOR KUTCHHI ENGINEERING COLLEGE.**

##### **LABORATORY**

**MANUAL**

**FOR**

**MICROPROCESSORS**

**AND**

**APPLICATIONS**

**FIND LARGEST & SMALLEST NUMBER IN AN ARRAY**

**Program Statement:**

Write a program to find out largest and smallest number in an array of 9 bytes stored at location 0600. Store the largest number at 0500H memory location.

**Apparatus:**

8086 based Microprocessor training kit

**Learning Objective:**

1. Comparison of numbers in the block.
2. Conditional jump instructions

**Logic:**

To find out largest and smallest number from block of given array initialize source index which address the memory location where data is stored. Initialize the counter. At start take byte of data from 0600 into AL register. Increment the SI. Compare the data in AL and data pointed SI. If data in AL is greater than data in memory location then decrement the counter and repeat the loop. If data in AL is smaller then move the data pointed by SI into AL, decrement the counter repeat the loop until counter becomes zero. Store the largest number into 0500H memory location.

**FIND LARGEST NUMBER IN AN ARRAY**

**Algorithm:**

1. Initialize SI as 0600H.
2. Initialize counter i.e. CX =0007H.
3. Move data pointed by SI into AL register.
4. Increment SI.
5. Compare data in AL and data pointed by SI.
6. If carry is NOT generated move the data pointed by SI into

AL registers.

1. Decrement counter.
2. If counter is not zero then go to step 3.
3. Initialize DI as 0500 and store largest number which is in

AL to location pointed by DI.

1. Stop.

**FINDING LARGEST NUMBER IN AN ARRAY**

Initialize memory pointer and counter

Increment SI

Copy content of location pointed by SI in AL

No

Yes

No

Yes

**FIND SMALLEST NUMBER IN AN ARRAY**

**Algorithm:**

1. Initialize SI as 0600H.
2. Initialize counter i.e. CX =0007H.
3. Move data pointed by SI into AL register.
4. Increment SI.
5. Compare data in AL and data pointed by SI.
6. If carry is generated move the data pointed by SI into

AL registers.

1. Decrement counter.
2. If counter is not zero then go to step 3.
3. Initialize DI as 0500 and store largest number which is in

AL to location pointed by DI.

1. Stop.

**FINDING SMALLEST NUMBER IN AN ARRAY**

Move data pointed by SI into AL Move data pointed by SI into AL

StartStart

Initialize memory pointer and counter

Copy content of location pointed by SI in AL

Increment SI

If number in AL< [SI]If number in AL> [SI]

Decrement CX Decrement CX

Is

CX=0?Is

CX=0?

DI=0500H

DI=0500H

Store content of AL to location pointed by DI

Store content of AL to location pointed by DI

StopStop

NO

YES

YES

NO

**FINDING LARGEST NUMBER IN AN ARRAY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory**  **Address** | **OP**  **CODE** | **Label** | **Mnemonic And Operand** | **Comment** |
| 01000 | BE |  | MOV SI, 0600H | Load SI with content of 0600H |
| 01001 | 00 |  |  |  |
| 01002 | 06 |  |  |  |
| 01003 | B9 |  | MOV CX, 0008H | Load CX = 8 |
| 01004 | 08 |  |  |  |
| 01005 | 00 |  |  |  |
| 01006 | 8A |  | MOV AL, [SI] | Load data in AL |
| 01007 | 04 |  |  |  |
| 01008 | 46 | AGAIN | INC SI | Increment SI Address |
| 01009 | 3A |  | CMP AL, [SI] | Compare |
| 0100A | 04 |  |  |  |
| 0100B | 73 |  | JNC NEXT | Jump if not Carry |
| 0100C | 02 |  |  |  |
| 0100D | 8A |  | MOV AL, [SI] | Move bigger no. to AI |
| 0100E | 04 |  |  |  |
| 0100F | 49 | NEXT | DEC CX | Decrement CX counter |
| 01010 | 75 |  | JNZ AGAIN | Jump if not set Zero flag |
| 01011 | F6 |  |  |  |
| 01012 | BF |  | MOV DI, 0500H | Move ans to 0500H |
| 01013 | 00 |  |  |  |
| 01014 | 05 |  | MOV [DI], AL |  |
| 01015 | 88 |  |  |  |
| 01016 | 05 |  |  |
| 01017 | F4 |  | HLT |  |

**FINDING LARGEST NUMBER IN AN ARRAY**

**Result:**

**Input: Output:**

|  |  |
| --- | --- |
| **Memory**  **Address** | **Data** |
| 0500 | 89 |

|  |  |
| --- | --- |
| **Memory**  **Address** | **Data** |
| 0600 | 32 |
| 0601 | 27 |
| 0602 | 55 |
| 0603 | 57 |
| 0604 | 89 |
| 0605 | 63 |
| 0606 | 01 |
| 0607 | 23 |
| 0608 | 52 |

**FINDING SMALLEST NUMBER IN AN ARRAY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory**  **Address** | **OP**  **CODE** | **Label** | **Mnemonic And Operand** | **Comment** |
| 01000 | BE |  | MOV SI, 0600H | Load SI with content of 0600H |
| 01001 | 00 |  |  |  |
| 01002 | 06 |  |  |  |
| 01003 | B9 |  | MOV CX, 0008H | Load CX = 8 |
| 01004 | 08 |  |  |  |
| 01005 | 00 |  |  |  |
| 01006 | 8A |  | MOV AL, [SI] | Load data in AL |
| 01007 | 04 |  |  |  |
| 01008 | 46 | AGAIN | INC SI | Increment SI Address |
| 01009 | 3A |  | CMP AL, [SI] | Compare |
| 0100A | 04 |  |  |  |
| 0100B | 73 |  | JNC NEXT | Jump if not Carry |
| 0100C | 02 |  |  |  |
| 0100D | 8A |  | MOV AL, [SI] | Move bigger no. to AI |
| 0100E | 04 |  |  |  |
| 0100F | 49 | NEXT | DEC CX | Decrement CX counter |
| 01010 | 75 |  | JNZ AGAIN | Jump if not set Zero flag |
| 01011 | F6 |  |  |  |
| 01012 | BF |  | MOV DI, 0500H | Move ans to 0500H |
| 01013 | 00 |  |  |  |
| 01014 | 05 |  | MOV [DI], AL |  |
| 01015 | 88 |  |  |  |
| 01016 | 05 |  |  |
| 01017 | F4 |  | HLT |  |

**FINDING SMALLEST NUMBER IN AN ARRAY**

**Result:**

**Input: Output:**

|  |  |
| --- | --- |
| **Memory**  **Address** | **Data** |
| 0500 | 01 |

|  |  |
| --- | --- |
| **Memory**  **Address** | **Data** |
| 0600 | 32 |
| 0601 | 27 |
| 0602 | 55 |
| 0603 | 57 |
| 0604 | 89 |
| 0605 | 63 |
| 0606 | 01 |
| 0607 | 23 |
| 0608 | 52 |

**FINDING LARGEST NUMBER IN AN ARRAY**

**Question Bank**

1. Explain compare instruction?

* COMPARE is an important instruction widely used in 8085 microprocessor. The 8085 instruction set has two types of Compare operations: Compare with accumulator (CMP) and Compare immediate with accumulator (CPI). This is a 1-byte instruction.

1. Explain JMP instruction.

* As you can see from this example JMP is able to transfer control both forward and backward. It can jump anywhere in current code segment (65,535 bytes). Unlike JMP instruction that does an unconditional jump, there are instructions that do a conditional jumps (jump only when some conditions are in act).

1. What are SI and DI?

* SI is called source index and DI is destination index. As the name follows, SI is always pointed to the source array and DI is always pointed to the destination.