 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No: 92201703058</b>

### **Experiment-7**

#### **AIM: To perform Branch Operations.**

#### **THEORY:**

There are two types of branches or jumps namely conditional and unconditional branches.

#### **Unconditional Branch Instructions**

The unconditional branches are those in which the program counter jumps to the label address provided within the instruction.

#### **Conditional Branch Instructions**

On the other hand, the conditional branches are those instructions whose execution is based on some condition. It checks one or more flag conditions and transfers the control to a new memory location. There are two types of jumps namely Far and Near. In the far jumps, the program counter jumps to the memory location which lies outside the current code segment whereas in near jumps, the IP points to the memory address inside the current code segment and that is why the CS register remains unchanged in near jumps. The conditional jumps are also near jumps. The syntax of these instructions is:


Opcode LABEL

It is a 2-byte instruction consisting of 1-Byte Opcode and 1-byte Label. Label value is between 00 and FF. It is sign-extended to 16-bits and added to the contents of IP register. The target address must be within the -128 to +127 bytes of IP. The flag conditions are checked depending upon the instruction, if they are true the program control is transferred to the memory address pointed by the IP register. If the condition is not satisfied, then the program continues in a sequential manner.

#### **List of 8086 Conditional Branch Instructions**

Table below shows the mnemonics of all the conditional branches instructions.

Instructions	Operation	Testing Condition
JA/JNBE	Jump if Above/Not Below or Equal	C=0 and Z=0
JAЕ/JNB/JNC	Jump if Above or Equal/ Jump if Not Below/Jump if No Carry	C=0
JB/JNAE	Jump if Below/ Jump if Not Above or Equal	C=1

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No: 92201703058</b>

JBE/JNA	Jump if Below or Equal/ Jump if Not Above	(C or Z) = 1
JC	Jump if Carry	C=1
JNC	Jump if Not Carry	C=0
JCXZ	Jump if the CX register=0	CX=0
JE/JZ	Jump if Equal/ Jump if Zero	Z=1
JG/JNLE	Jump if Greater/ Jump if Not Less Than or Equal	((S xor O) or Z) = 0
JGE/JNL	Jump if Greater or Equal/ Jump if Not Less Than	(S xor O)=0
JL/JNGE	Jump if Less Than/ Jump if Not Greater Than or Equal	(S xor O)=1
JLE/JNG	Jump if Less than or Equal/ Jump if Not Greater	((S xor O) or Z) = 1
JNE/JNZ	Jump if Not Equal/ Jump if Not Zero	Z=0
JNP/JPO	Jump if Not Parity/ Jump if Parity Odd	P=0
JNS	Jump if Not Signed/ Jump if Positive	S=0
JO	Jump if Overflow	O=1
JNO	Jump if Not Overflow	O=0
JP/JPE	Jump if Parity/ Jump if Parity Even	P=1
JS	Jump if Signed/ Jump if Negative	S=1

These instructions are executed after some other instructions which affects the content of flag registers. Let's discuss these instructions in detail through examples.

### 8086 Microprocessor Assembly Comparison Example


The code below compares two numbers and print if number 1 is equal, greater or less than number 2. This code is implemented using three conditional branches which are JE, JB and JA.

#### Assembly Code

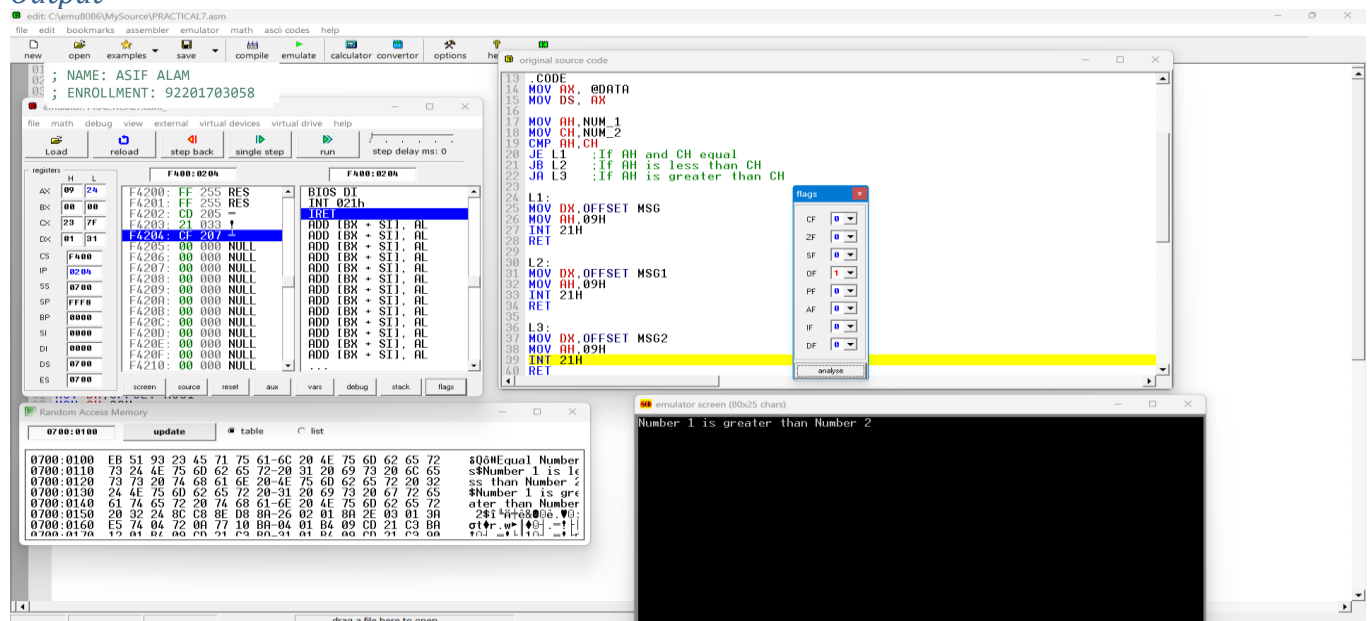
JA/JNBE will check the CF and ZF flags. If both are 0, then the IP will jump to the target address. Also, if both CF and ZF are equal to 1, then the program will continue to execute sequentially. Moreover, if any one of them is 0 and other is 1, then this instruction will have no effect of program execution.

#### Code:

The JA instruction will check if CF is 0. Even if you don't use this instruction in the code given below, it will not affect the program execution as if the first two conditions are not satisfied, then it means the number 1 is greater than number 2.


 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No: 92201703058</b>

## Output



If the two numbers are equal, zero flag (ZF) will be equal to 1. JE instruction checks ZF. If it is 1, the program counter jumps to L1 which will print the message "Equal numbers" on the emulator screen. If the zero flag is 0, then the program counter will jump to the next instruction which is JB L2. The JB or JBNE instructions check whether

CF flag is 1 or not. If it is 1, the control is transferred to label address. The Compare instruction subtracts the content of CH from AH. If AH is less than the CH, then it will borrow carry thus setting CF to 1. If CF is 1, then the instructions after label address L2 will undergo execution.

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No: 92201703058</b>

### 8086 Check Carry Flag :

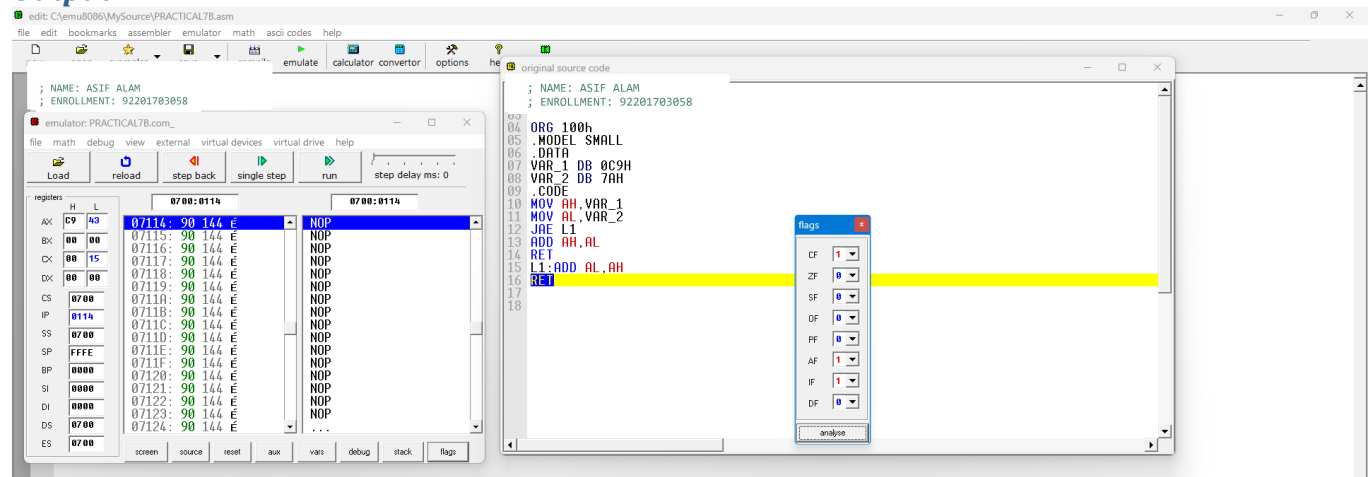
The JAE/JNB/JNC instructions check Carry flag (CF). If it is 0, jump to the target address. For example:


```
ADD AH, CH
JAE L1
```

Suppose AH=C9H and BH=7AH. The first instruction adds C9 and 7AH and gives 143. As it generates carry, therefore, CF becomes equal to 1. Now after the execution of JAE instruction, the program counter will not jump to L1 instead it would execute the next instruction after JAE instruction. Now replace ADD instruction with CMP instruction. If AH is greater or equal to CH data, then the program counter will jump to the L1 label.

### Code:

### Output



 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No: 92201703058</b>

### 8086 JO and JNO Branch Instruction:

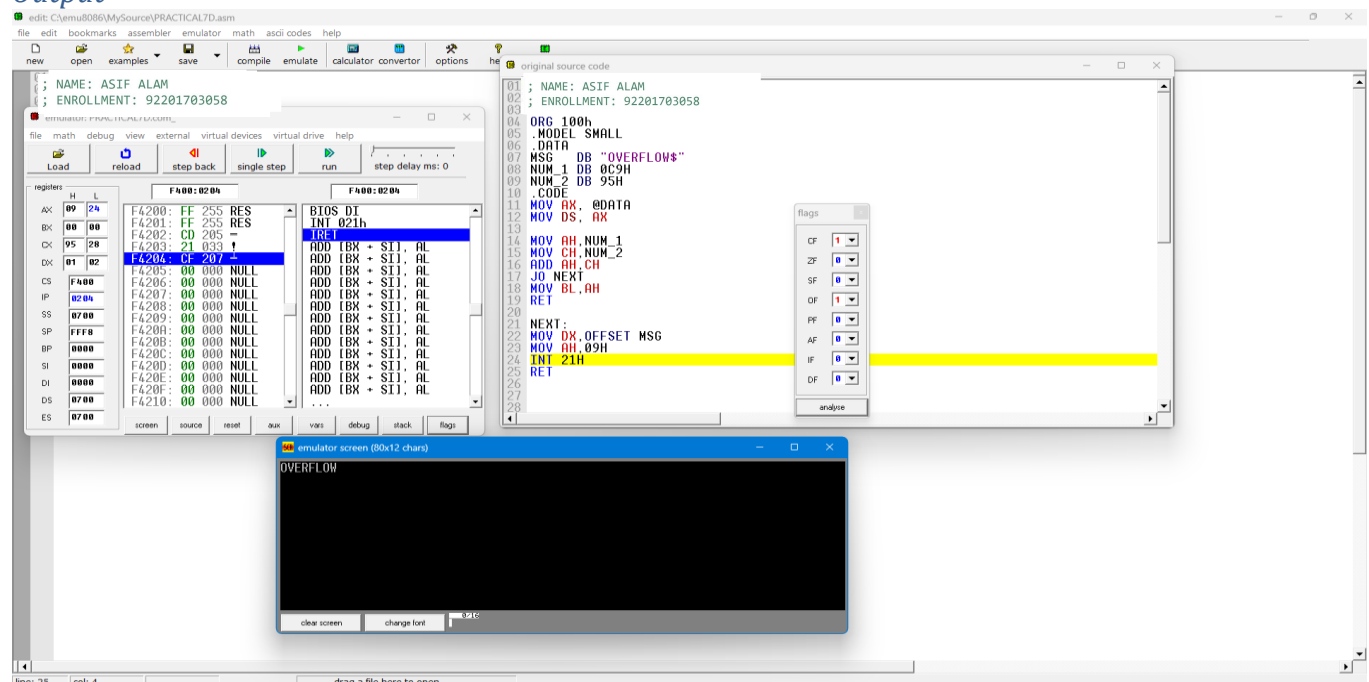
The code below explains the behavior of JO instruction. It adds two numbers and check the overflow. If the result is too large to fit in the destination register, then it will set overflow bit to 1. The JO instruction checks the overflow flag. If it is 1, the control will be transferred to NEXT label which will then display message of “OVERFLOW” on the emulator screen. If the OF=0, then the sum is stored in the variable “RESULT” which is stored in memory.


In this example, the two numbers are 0C9H and 95H. Adding these two numbers give 15E resulting into overflow. As OF = 1, therefore after JO instruction, the instructions after label NEXT are executed.

### Code

The JNO instruction is opposite to JO instruction. When OF is 0, JNO instruction will jump the program counter to the instruction whose label address is provided within the JNO instruction.

### Output



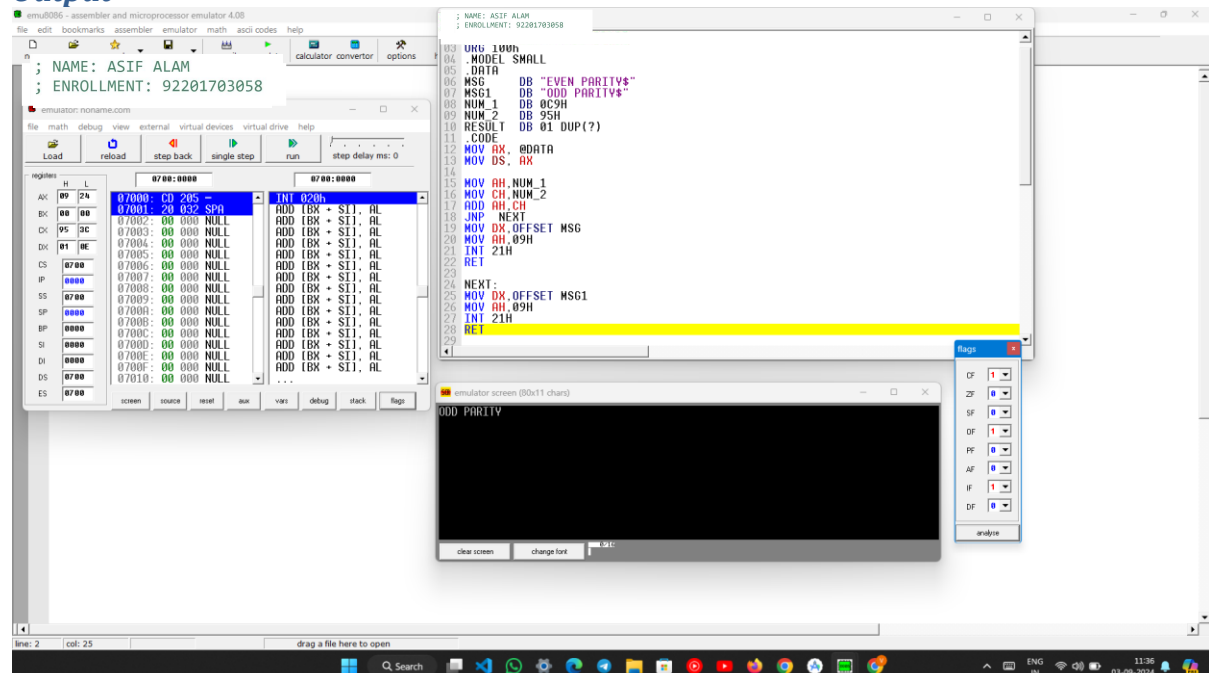
 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No: 92201703058</b>


## 8086 JNP Branch Instruction:

The JNP instruction checks the parity flag. If parity is odd or PF=0, the program counter will jump to the label address. The JP instruction checks if parity is even or PF=1.

## Code

## Output



 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No:</b> 92201703058

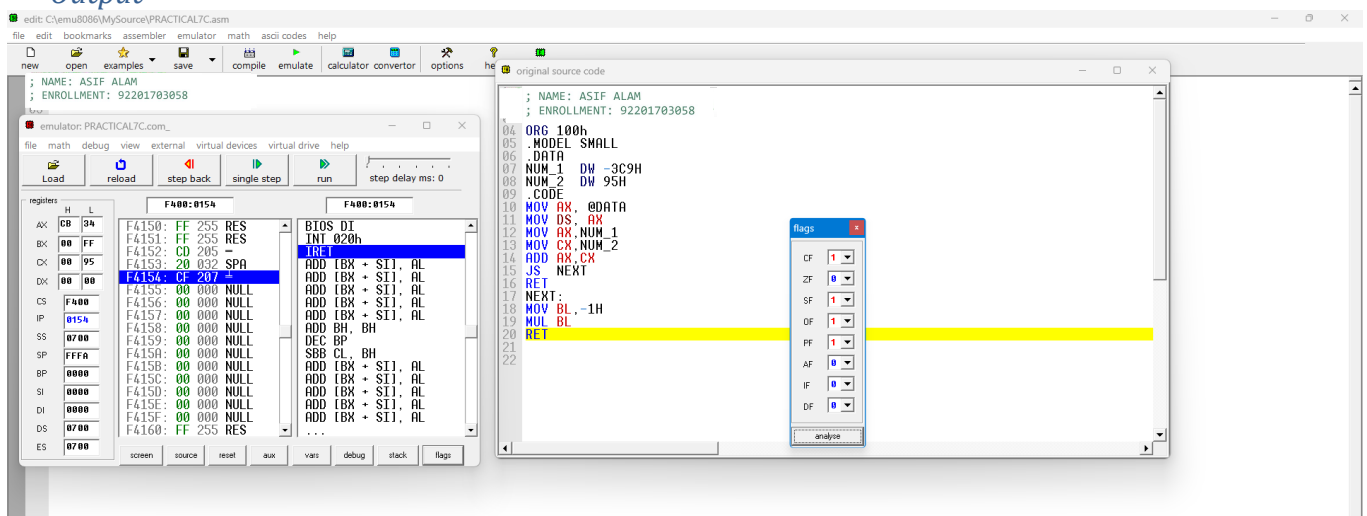
## 8086 JNS Branch Instruction:



There is another instruction that is used to check whether the result is positive or negative and jumps to the label address depending upon the sign of the result. The JNS instruction is a jump if the result of the previous instruction is not a signed number or positive whereas JS instruction is a jump if the previous instruction is a signed value or negative.

## Assembly Code

The code in example 5 adds two numbers and check if the result is a positive number or not. If the number is positive, the program stops running otherwise the program will jump to the NEXT label in which the result is multiplied with -1 to convert it into a positive number.

## Output



 <b>Marwadi University</b> <small>Marwadi Chandarana Group</small> 	<b>Marwadi University</b> <b>Department of Computer Engineering</b>	
<b>Subject: Fundamental of Processors (01CE0509)</b>	<b>Aim: To perform Branch Operations.</b>	
<b>Experiment No: 07</b>	<b>Date:</b>	<b>Enrolment No:</b> 92201703058

**Conclusion :**