Computer Organization and Assembly Language **Instructor:** *Dr. Muhammad Usama*
Assignment 1
15 September 2022

---

1. Sort the following data in ascending order using the bubble sort algorithm: ***num1: dw 60, 55, 45, 50, 40, 35, 25, 30, 10, 0***. Give a written dry run of the code and also provide a screen capture of the AFD debugger. Also, defend your choice of branches. **(25 Points)**

2. Sort the following data in descending order using the bubble sort algorithm: ***num2: dw 60, 55, 45, 50, -40, -35, 25, 30, 10, 0***. Give a written dry run of the code and also provide a screen capture of the AFD debugger. Also, defend your choice of branches. **(25 Points)**

3. Sort the following data in ascending order using the bubble sort algorithm: ***num3: 60, 55, 55, 60, 40,40, 59, 15, 25, 15***. Give a written dry run of the code and also provide a screen capture of the AFD debugger. Also, defend your choice of branches. **(25 Points)**

4. Sort the following data in descending order using the bubble sort algorithm: ***num4: 60, 55, -55, 60, 60, 58, -58, -58, 25, 15***. Give a written dry run of the code and also provide a screen capture of the AFD debugger. Also, defend your choice of branches. **(25 Points)**

		positive number signals that the destination is smaller.
SDEST ≤ SSRC	ZF = 1 OR SF ≠ OF	If the zero flag is set, it means that the source and destination are equal and if the sign and overflow flags differ it means that the destination is smaller as described above.
SDEST ≥ SSRC	SF = OF	When a signed source is subtracted from a signed destination and the answer is positive with no overflow than the destination is greater than the source. When an overflow is there signaling that sign has changed unexpectedly, we interpret a negative answer as the signal that the destination is greater.
SDEST > SSRC	ZF = 0 AND SF = OF	If the zero flag is not set, it means that the signed operands are not equal and if the sign and overflow match in addition to this it means that the destination is greater than the source.

3.2. CONDITIONAL JUMPS

For every interesting or meaningful situation of flags, a conditional jump is there. For example JZ and JNZ check the zero flag. If in a comparison both operands are same, the result of subtraction will be zero and the zero flag will be set. Thus JZ and JNZ can be used to test equality. That is why there are renamed versions JE and JNE read as jump if equal or jump if not equal. They seem more logical in writing but mean exactly the same thing with the same opcode. Many jumps are renamed with two or three names for the same jump, so that the appropriate logic can be conveyed in assembly language programs. This renaming is done by Intel and is a standard for iAPX88. JC and JNC test the carry flag. For example we may need to test whether there was an overflow in the last unsigned addition or subtraction. Carry flag will also be set if two unsigned numbers are subtracted and the first is smaller than the second. Therefore the renamed versions JB, JNAE, and JNB, JAE are there standing for jump if below, jump if not above or equal, jump if not below, and jump if above or equal respectively. The operation of all jumps can be seen from the following table.

JC JB JNAE	Jump if carry Jump if below Jump if not above or equal	CF = 1	This jump is taken if the last arithmetic operation generated a carry or required a borrow. After a CMP it is taken if the unsigned source is smaller than the unsigned destination.
JNC JNB JAE	Jump if not carry Jump if not below Jump if above or equal	CF = 0	This jump is taken if the last arithmetic operation did not

			generated a carry or required a borrow. After a CMP it is taken if the unsigned source is larger or equal to the unsigned destination.
JE JZ	Jump if equal Jump if zero	ZF = 1	This jump is taken if the last arithmetic operation produced a zero in its destination. After a CMP it is taken if both operands were equal.
JNE JNZ	Jump if not equal Jump if not zero	ZF = 0	This jump is taken if the last arithmetic operation did not produced a zero in its destination. After a CMP it is taken if both operands were different.
JA JNBE	Jump if above Jump if not below or equal	ZF = 0 AND CF = 0	This jump is taken after a CMP if the unsigned source is larger than the unsigned destination.
JNA JBE	Jump if not above Jump if not below or equal	ZF = 1 OR CF = 1	This jump is taken after a CMP if the unsigned source is smaller than or equal to the unsigned destination.
JL JNGE	Jump if less Jump if not greater or equal	SF ≠ OF	This jump is taken after a CMP if the signed source is smaller than the signed destination.
JNL JGE	Jump if not less Jump if greater or equal	SF = OF	This jump is taken after a CMP if the signed source is larger than or equal to the signed destination.
JG JNLE	Jump if greater Jump if not less or equal	ZF = 0 AND SF = OF	This jump is taken after a CMP if the signed source is larger than the signed destination.
JNG JLE	Jump if not greater Jump if less or equal	ZF = 1 OR SF ≠ OF	This jump is taken after a CMP if the signed source is smaller than or equal to the signed destination.

JO	Jump if overflow.	OF = 1	This jump is taken if the last arithmetic operation changed the sign unexpectedly.
JNO	Jump if not overflow	OF = 0	This jump is taken if the last arithmetic operation did not change the sign unexpectedly.
JS	Jump if sign	SF = 1	This jump is taken if the last arithmetic operation produced a negative number in its destination.
JNS	Jump if not sign	SF = 0	This jump is taken if the last arithmetic operation produced a positive number in its destination.
JP JPE	Jump if parity Jump if even parity	PF = 1	This jump is taken if the last arithmetic operation produced a number in its destination that has even parity.
JNP JPO	Jump if not parity Jump if odd parity	PF = 0	This jump is taken if the last arithmetic operation produced a number in its destination that has odd parity.
JCXZ	Jump if CX is zero	CX = 0	This jump is taken if the CX register is zero.

The CMP instruction sets the flags reflecting the relation of the destination to the source. This is important as when we say jump if above, then what is above what. The destination is above the source or the source is above the destination.

The JA and JB instructions are related to unsigned numbers. That is our interpretation for the destination and source operands is unsigned. The 16th bit holds data and not the sign. In the JL and JG instructions standing for jump if lower and jump if greater respectively, the interpretation is signed. The 16th bit holds the sign and not the data. The difference between them will be made clear as an elaborate example will be given to explain the difference.

One jump is special that it is not dependant on any flag. It is JCXZ, jump if the CS register is zero. This is because of the special treatment of the CX register as a counter. This jump is regardless of the zero flag. There is no counterpart or not form of this instruction.

The adding numbers example of the last chapter can be a little simplified using the compare instruction on the BX register and eliminating the need for a separate counter as below.