

Short Notes Chapter 3 [Lecture 8-9]

CMP:

The basic root instruction for all comparisons is CMP standing for compare. The operation of CMP is to subtract the source operand from the destination operand, updating the flags without changing either the source or the destination. CMP is meaningless without a conditional jump immediately following it.

Jumps for Signed and Unsigned numbers:

For signed numbers JG and JL will work properly and for unsigned JA and JB will work properly and not the other way around.

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 unsigned comparisons see the numbers as 0 being the smallest and 65535 being the largest with the order that $0 < 1 < 2 \dots < 65535$. The signed comparisons see the number -32768 which has the same memory representation as 32768 as the smallest number and 32767 as the largest with the order $-32768 < -32767 < \dots < -1 < 0 < 1 < 2 < \dots < 32767$

All meaningful situations both for signed and unsigned numbers that occur after a comparison are detailed in the following table.

DEST = SRC	ZF = 1	When the source is subtracted from the destination and both are equal the result is zero and therefore the zero flag is set. This works for both signed and unsigned numbers.
UDEST < USRC	CF = 1	When an unsigned source is subtracted from an unsigned destination and the destination is smaller, borrow is needed which sets the carry flag.
UDEST <= USRC	ZF = 1 OR CF = 1	If the zero flag is set, it means that the source and destination are equal and if the carry flag is set it means a borrow was needed in the subtraction and therefore the destination is smaller.

UDEST>=USRC	CF = 0	When an unsigned source is subtracted from an unsigned destination no borrow will be needed either when the operands are equal or when the destination is greater than the source.
UDEST > USRC	ZF = 0 AND CF = 0	The unsigned source and destination is not equal if the zero flag is not set and the destination is not smaller since no borrow was taken. Therefore, the destination is greater than the source.
SDEST < SSRC	SF≠OF	When a signed source is subtracted from a signed destination and the answer is negative with no overflow than the destination is smaller than the source. If, however there is an overflow meaning that the sign has changed unexpectedly, the meanings are reversed and a 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.

Conditional Jumps:

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 destination is smaller than the unsigned source.
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 generate a carry or required a borrow . After a CMP it is taken if the unsigned destination is larger or equal to the unsigned source.
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 produce 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 destination is larger than the unsigned source.
JNA JBE	Jump if not above Jump if below or equal	ZF = 1 OR CF = 1	This jump is taken after a CMP if the unsigned destination is smaller than or equal to the unsigned source.

SSWT [SUPERSTARWEBTECH.COM]
 CS401-ASSEMBLY LANGUAGE AND PROGRAMMING
 CS401 CHAPTER NO 3 [LECTURE 8-9]
 PREPARED BY: [HTTPS://WWW.SUPERSTARWEBTECH.COM/](https://www.superstarwebtech.com/)

JL JNGE	Jump if less Jump if not greater or equal	SF≠OF	This jump is taken after a CMP if the signed destination is smaller than the signed source.
JNL JGE	Jump if not less Jump if greater or equal	SF = OF	This jump is taken after a CMP if the signed destination is larger than or equal to the signed source.
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 destination is larger than the signed source.
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 destination is smaller than or equal to the signed source.
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.

One jump is special that it is not dependent on any flag. It is JCXZ, jump if the CX 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.

Unconditional Jump:

A new instruction called JMP. It is the unconditional jump that executes regardless of the state of all flags

Types of Jumps:

The three types of jump, near, short, and far, differ in the size of instruction and the range of memory they can jump to with the smallest short form of two bytes and a range of just 256 bytes to the far form of five bytes and a range covering the whole memory.

Near Jump

When the relative address stored with the instruction is in 16 bits, the jump is called a near jump. Using a near jump we can jump anywhere within a segment.

Short Jump

If the offset is stored in a single byte as in 75F2 with the opcode 75 and operand F2, the jump is called a short jump. F2 is added to IP as a signed byte. If the byte is negative the complement is negated from IP otherwise the byte is added. Unconditional jumps can be short, near, and far.

Conditional jumps can only be short. A short jump can go +127 bytes ahead in code and -128 bytes backwards and no more. This is the limitation of a byte in signed representation.

Far Jump

Far jump is not position relative but is absolute. Both segment and offset must be given to a far jump. The previous two jumps were used to jump within a segment. Sometimes we may need to go from one code segment to another, and near and short jumps cannot take us there. Far jump must be used and a two-byte segment and a two-byte offset are given to it. It loads CS with the segment part and IP with the offset part.

Execution therefore resumes from that location in physical memory. The three instructions that have a far form are JMP, CALL, and RET, are related to program control. Far capability makes intra segment control possible.