Assignment 3
Gunnar Yonker

1. lea:

The lea is the Load Effective Address instruction in x86 assembly language, and this instruction is used to load the effective address of a memory location into a register. When used it will calculate the offset address of a memory location and load it into a register without accessing the memory location. The lea instruction is most often used to perform arithmetic operations and address calculations.

Syntax:

lea dest, src

The lea instruction takes two operands: the destination and the source. The source operand specifies a memory location or an address expression that is used to calculate the effective address. The destination operand specifies a register that will hold the calculated effective address. The source operand can be an immediate value, a memory address, a register, or a combination of these. The destination operand can only be a register.

Example:

lea eax, [ebx+4]

The above instruction will load the effective address of the memory location [ebx+4] into the eax register.

2. leave:

The leave instruction is used to deallocate a stack frame that was created when the enter instruction was used. When used it will copy the base pointer (BP) value to the stack pointer (SP) and will then put the previous value of the base pointer from the stack. This instruction is most often used at the end of a subroutine to restore the calling function's stack frame.

Syntax:

leave

The leave instruction does not take any operands.

Example:

enter 0, 0

; code here that sets up a stack frame

leave

ret

The above instruction shows how leave could be used. In this example the enter instruction sets up a stack frame which includes allocating space for local variables and saving the base pointer on the stack. The leave instruction is then used to deallocate the stack frame by restoring the previous value of the base pointer and adjusting the stack pointer.

Assignment 3
Gunnar Yonker

3. xor:

The xor is the Exclusive OR instruction in x86 assembly language and it performs a bitwise XOR operation between two operands and then stores the result in the destination operand. Also, the XOR operation sets each bit of the destination operand to 1 if the corresponding bits of the two operands are different, and if they are not it is set to 0. Most often the xor instruction is used to clear a register or memory location.

Syntax:

xor dest, src

The xor instruction takes two operands: destination and source. Both operands can be registers, memory location, or immediate values. The destination operand stores the result of the XOR operation, the source operand is not modified by the xor instruction.

Example:

xor eax, eax

The above instruction example will set the eax register to zero by performing a bitwise XOR operation between eax and itself. It is mainly used to clear a register or memory location.

4. pop:

The pop instruction is used to remove the topmost 4-byte data element value from the stack and store it in a specified register or memory location. It will also adjust the stack pointer to remove the popped value from the stack.

Syntax:

pop dest

The pop instruction takes one operand which is the destination of the popped value. The destination operand can be a register or a memory location.

Example:

pop eax

The pop instruction example above will remove the topmost value from the stack and store it into the eax register.

Assignment 3
Gunnar Yonker

5. movsb:

The movsb instruction is used to move a byte of data from the source address to the destination address that is specified. It is most often used to perform memory copies and string operations.

Syntax:

```
movsb
```

The movsb instruction does not take any operands. The source address is pointed to by the destination index register. After the move, both the source index and destination index registers are automatically incremented or decremented depending on the direction of the move.

Example:

```
movsb
```

When the movsb instruction is used, it moves a single byte of data from the memory location pointed to by the source index register to the memory location that is pointed to by the destination index register. The source and destination addresses are implicitly determined by the values in the SI and DI registers.