Instruction	Definition	Usage Example
MOV	Move data from one location to another.	`mov eax, ebx` moves the contents of the EBX register into the EAX register.
ADD	Add two operands and store the result.	`add eax, 10` adds the immediate value 10 to the contents of the EAX register.
SUB	Subtract two operands and store the result.	`sub eax, ebx` subtracts the contents of the EBX register from the EAX register.
CMP	Compare two operands and set the appropriate flags.	`cmp eax, ebx` compares the contents of the EAX and EBX registers and sets the appropriate flags.
JMP	Unconditionally jump to a new location in the code.	`jmp label` jumps to the code label.
JZ/JNZ	Jump if the zero flag is set or not set, respectively.	`jz label` jumps to the code label if the zero flag is set.
JE/JNE	Jump if two operands are equal or not equal, respectively.	`je label` jumps to the code label if the operands are equal.
CALL	Call a subroutine or function.	`call my_function` calls the function named my_function.
RET	Return from a subroutine or function.	`ret` returns from the current function.
PUSH	Push a value onto the stack.	`push eax` pushes the contents of the EAX register onto the stack.
POP	Pop a value from the stack into a register.	`pop ebx` pops the top value from the stack into the EBX register.
XOR	Perform a bitwise exclusive OR operation on two operands.	`xor eax, ebx` performs a bitwise XOR operation on the contents of the EAX and EBX registers.
AND	Perform a bitwise AND operation on two operands.	`and eax, ebx` performs a bitwise AND operation on the contents of the EAX and EBX registers.
OR	Perform a bitwise OR operation on two operands.	`or eax, ebx` performs a bitwise OR operation on the contents of the EAX and EBX registers.
NOT	Perform a bitwise NOT operation on an operand.	`not eax` performs a bitwise NOT operation on the contents of the EAX register.
SHL	Shift the bits of an operand left by a specified number of bits.	`shl eax, 1` shifts the contents of the EAX register left by one bit.
SHR	Shift the bits of an operand right by a specified number of bits.	`shr ebx, 3` shifts the contents of the EBX register right by three bits.