Operand Extension and Address Extension

Operand Extension

Most CPUs operate on fixed-size data words (e.g., 32-bit or 64-bit). However, instruction operands may be shorter (e.g., 8-bit or 16-bit), especially in RISC architectures. These shorter operands need to be extended to match the CPU word size before operations can be performed. There are two primary techniques for operand extension:

- **Sign Extension**
- **Zero Extension**

Sign Extension

Used for **signed numbers** (represented in two's complement). It preserves the number's sign and magnitude by replicating the sign bit.

Procedure:

- Let N be an m-bit signed number.
- Let $n > m$ be the target word size.
- Replicate the most significant bit (MSB), also the sign bit s, $(n - m)$ times to the left.

Example:

$$N = \texttt{10101 01010101} \quad \text{(13-bit)}$$

$$\text{Sign-Extended } N = \underbrace{\texttt{1111111111111111111}}_{n-m=19 \text{ bits}}\texttt{010101010101}$$

$$N = -2731_{10}$$

This method ensures that the extended value maintains the correct signed interpretation.

Zero Extension

Used for **unsigned numbers**. The value is extended by adding leading zero bits.

Procedure:

- Let N be an m-bit unsigned number.

- Extend it to n bits by prefixing $(n - m)$ zeros.

Example:

$$N = \texttt{10101 01010101} \quad \text{(13-bit)}$$
$$\text{Zero-Extended } N = \underbrace{\texttt{0000000000000000000}}_{n-m=19 \text{ bits}}\texttt{010101010101}$$

The numerical value remains unchanged, only the size increases.

Address Extension

In many architectures, instruction formats limit the number of bits available for addressing. This causes the address field in instructions to be shorter (e.g., m-bits) than the full memory address width (n-bits).

The Problem

If a short address is directly zero-extended, only 2^m memory locations can be accessed. This limitation restricts the memory range.

The Solution: Base + Offset Addressing

To overcome the addressing limitation:

- Treat the m-bit address as an **offset**.
- Store a full-size base address (n-bits) in a CPU register.
- Compute the effective memory address as:

$$\text{Effective Address} = \text{Base Register} + \text{Zero-Extended Offset}$$

This technique allows addressing a larger memory range and is widely used in both RISC and CISC processors.

Instruction Example

$$\texttt{STB Rs, Rd(S2)} \Rightarrow M[\texttt{Rd + S2}] := \texttt{Rs[24:31]}$$

Explanation:

- `S2` is a short offset (e.g., 13-bit).
- `Rd` holds the base address (e.g., 32-bit).
- The effective address is computed as `Rd + Zero-Extended(S2)`.
- A byte from `Rs` is stored at that memory location.

Summary Table

Extension Type	**Used For**	**Method**	**Example**
Sign Extension	Signed Numbers	Replicate sign bit to left	1010101010101 → 111111...0101010
Zero Extension	Unsigned Numbers	Add leading zeros	1010101010101 → 000000...0101010
Address Extension	Short address field	Zero-extend + add base	Effective Address = Base + Offset