Addressing Modes Revisited

I disagree slightly with the author of the text we are using on the names for certain of the 80x86 addressing modes. The author uses the following names for certain of the addressing modes:

[BX], [SI], [DI], [BP]  Register Indirect
[BX+DISP], [BP+DISP]  Based
[SI+DISP], [DI+DISP]  Indexed

This is almost consistent with the Intel documentation for processors up through the 80286. It is inconsistent in that Intel doesn’t call [BP] addressing register indirect.

I prefer to use the following names for the addressing modes:

[BX], [BP]  Based
[SI], [DI]  Indexed
[BX+DISP], [BP+DISP]  Base+Displacement
[SI+DISP], [DI+DISP]  Index+Displacement

I think that these names are more descriptive, and are also more consistent with the Intel documentation for the 80386 and later processors.

The reason that I prefer the second set of names for the above addressing modes is that I think it makes it easier to remember which registers can be used in which ways. This is particularly important when it comes to which of the combinations are valid when using BASE+INDEX and BASE+INDEX+DISPLACEMENT addressing. It makes it easier to remember which registers can be combined to form an address.

Memory addressing modes can be considered to be in a kind of hierarchy. According to the following breakdown:
Direct - The instruction specifies the address
Indirect - The instructions specifies where to find the address.

There is really only one kind of direct addressing. The instruction directly specifies the address of the memory data. However, there are several possible types of indirect addressing. These fall into two general types or categories:

- Memory Indirect - A memory location contains the address
- Register Indirect - A register contains the address

To be generally useful, it is necessary for a computer to be able to operate on higher-level data structures such as arrays and lists. In order to access array or list data, it is necessary to be able to compute an address at run time. This implies the need for a way to use the computed address to access memory, and hence the need for some form of indirect addressing.

Essentially, all CPU’s support some form of indirect addressing and it is generally a register indirect addressing mode. Not many machines implement any form of memory indirect addressing. The Data General Nova and Eclipse mini-computers, for example, had a memory indirect addressing mode, but most machines don’t.

According to the above taxonomy, Register Indirect addressing, when viewed in the context of the 80x86 architecture can be considered to be a class of addressing modes. Within the general category of Register Indirect, the following addressing modes exist in the 80x86 processors:

- Based
  A base register specifies the address (BX or BP)
- Base+Displacement
  A base register plus a constant displacement specifies the address
- Indexed
  An index register specifies the address (SI or DI)
- Index+Displacement
  An index register plus a constant displacement specifies the address
Base+Index  The sum of a base register and an index registers specifies the address

Base+Index+Displacement  The sum of a base register, an index register and a constant displacement specifies the address.

The following is a list of the memory addressing modes, and my preferred names for them. Also included, in parenthesis, are the textbook’s names for these addressing modes.

- **[BX] or [BP]** Based (Register Indirect)
- **[BX+DISP] or [BP+DISP]** Base+Displacement (Based)
- **[SI] or [DI]** Index (Register Indirect)
- **[SI+DISP] or [DI+DISP]** Index+Displacement (Indexed)
- **[BX+SI] or [BP+SI]** Base+Index
- **[BX+DI] or [BP+DI]** Base+Index
- **[BX+SI+DISP] or [BP+SI+DISP]** Base+Index+Displacement
- **[BX+DI+DISP] or [BP+DI+DISP]** Base+Index+Displacement

The above list gives all possible valid indirect memory addressing modes on the 8088/8086 processors. Beginning with the 80386 processor, other indirect memory addressing modes are supported, but we are not considering them here.

**Masm Addressing Mode Syntax**

The processor itself supports certain addressing modes in the instruction set. In order to specify an addressing mode to be used, the assembly language programmer must code a statement using some syntax acceptable to the assembler that indicates the addressing mode to be used. **MASM supports several alternative syntaxes that can be use to specify an addressing mode.**
In general, placing a register inside of square brackets ([]) indicates that the contents of that register should be used as an address. Referencing a register name outside of brackets indicates that the content of the register itself is to be used.

For based and indexed addressing (without displacement) where only a single element (the contents of the base or index register) forms the address, this is sufficient, and there are no alternative syntaxes. The complexity enters when two or more elements come together to form the address (e.g. index+displacement). In this case there are several ways in which the addressing mode can be specified.

For example:

- [si] - indexed addressing using contents of SI
- [si+2] - indexed+displacement
- foo[si] - indexed+displacement
- foo[si+2] - indexed+displacement
- [si+foo] - indexed+displacement
- [si+foo+2] - indexed+displacement
- foo[si].elem - indexed+disp (structure member reference)
- [si+cbfoo*8+4] - indexed+displacement
- [bx+si] - base + index
- [bx][si] - base + index
- foo[bx][si] - base + index + displacement
- [bx+2][si+foo] - base + index + displacement
- foo[bx][si+2] - base + index+displacement

In the indexed+displacement or based+displacement addressing modes, the value in the index register or base register is added to a constant and this sum forms the effective address of the memory operand. The key point is that the displacement must be a constant. Syntactically, MASM will allow any expression containing only constants and will evaluate the expression at assembly time, resolving it to a single constant. Remember that variable names are symbolic constants.

The name of a variable is a symbolic constant whose value is the address of the variable. Thus, a variable name can be used anywhere
a constant would be used. Whenever a constant is adjacent to a square bracket containing an index or base register, there is an addition implied, (e.g. foo[si]).