x86 Addressing Modes

The addressing mode determines, for an instruction that accesses a memory location, how the address for the memory location is specified.

To summarize, the following diagram (from http://en.wikipedia.org/wiki/X86#Addressing_modes) shows the possible ways an address could be specified:

$$\left[\begin{Bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ ESP \\ EBP \\ ESI \\ EDI \end{Bmatrix}\right] + \left[\begin{Bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ EBP \\ ESI \\ EDI \end{Bmatrix} * \begin{Bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{Bmatrix}\right] + [\text{displacement}]$$

Each square bracket in the above diagram indicates an optional part of the address specification. These parts (from left to right) are: A register used as a base address, a register used as an index, a width (or scale) value to multiply the register by, and an displacement (aka offset) which is an integer. The address is computed as the sum of: the base register, the index times the width, and the displacement.

The Intel & AT&T syntax for various addressing modes, depending on which parts of the above diagram are used, is shown in the table below from http://simon.baymoo.org/universe/tools/symset/symset.txt (slightly modified):

Mode	Intel	AT&T
Absolute	MOV EAX, [0100]	movl 0x0100, %eax
Register	MOV EAX, [ESI]	movl (%esi), %eax
Reg + Off	MOV EAX, [EBP-8]	movl -8(%ebp), %eax
R*W + Off	MOV EAX, [EBX*4 + 0100]	movl 0x100(,%ebx,4), %eax
B + R*W + O	MOV EAX, [EDX + EBX*4 + 8]	movl 0x8(%edx,%ebx,4), %eax

Note that, given the definition of a label `x` in the `.data` section of an assembly program, using x to indicate the memory location, as in

```
mov  eax,x  #Intel
```

or

```
mov  x,%eax  #AT&T
```

is just absolute addressing (i.e. using just a displacement), where the assembler essentially replaces the name `x` with the address corresponding to `x`.