Addressing Modes

- **Implied -** the data value/data address is implicitly associated with the instruction.
- **Register** - references the data in a register or in a register pair.
- **Immediate -** the data is provided in the instruction.
- **Direct** - the instruction operand specifies the memory address where data is located.
- **Register indirect -** instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.
- **Based** :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides.

Next Page

- **Indexed** :- 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
- **Based Indexed** :- the contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
- **Based Indexed with displacement** :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.

Memory

- Program, data and stack memories occupy the same memory space. As the most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory.
- To access memory outside of 64 KB the CPU uses special segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory (see the "Registers" section below).
- 16-bit pointers and data are stored as:
  address: low-order byte
  address+1: high-order byte

Next Page

- 32-bit addresses are stored in "segment: offset" format as:
  address: low-order byte of segment
  address+1: high-order byte of segment
  address+2: low-order byte of offset
  address+3: high-order byte of offset

- Physical memory address pointed by segment: offset pair is calculated as:
- address = (<segment> * 16) + <offset>

Next Page

- **Program memory** - program can be located anywhere in memory. Jump and call instructions can be used for short jumps within currently selected 64 KB code segment, as well as for far jumps anywhere within 1 MB of memory.
- All conditional jump instructions can be used to jump within approximately +127 to -127 bytes from current instruction.
- **Data memory** - the processor can access data in any one out of 4 available segments, which limits the size of accessible memory to 256 KB (if all four segments point to different 64 KB blocks).

Next Page

- Accessing data from the Data, Code, Stack or Extra segments can be usually done by prefixing instructions with the DS:, CS:, SS: or ES: (some registers and instructions by default may use the ES or SS segments instead of DS segment).
- Word data can be located at odd or even byte boundaries. The processor uses two memory accesses to read 16-bit word located at odd byte boundaries. Reading word data from even byte boundaries requires only one memory access.

Next Page

- **Stack memory** can be placed anywhere in memory. The stack can be located at odd memory addresses, but it is not recommended for performance reasons (see "Data Memory" above).

Reserved locations:

- 0000h - 03FFh are reserved for interrupt vectors. Each interrupt vector is a 32-bit pointer in format segment: offset.
- FFFF0h - FFFFFh - after RESET the processor always starts program execution at the FFFF0h address.