Align Directive and PTR Byte

Used to align a variable or label on a specific memory boundary, such as a byte, word, doubleword, or paragraph boundary.

The syntax for the ALIGN directive is as follows: ALIGN bound. For example, **ALIGN 4** aligns the variable or label on a **4-byte boundary.**

Using PTR with Data Types:

When moving data between memory and registers in assembly language, the PTR operator is used in combination with standard assembler data types.

These data types include BYTE, SBYTE, WORD, SWORD, DWORD, SDWORD, FWORD, QWORD, or TBYTE. The PTR operator is used to specify the size of the data being moved and helps ensure proper alignment.

Moving Smaller Values into Larger Destinations:

In some cases, you may need to move smaller values from memory into a larger destination operand, such as moving a word into a doubleword. This can be achieved using the DWORD PTR operator. For example:

```
.data
    wordList WORD 5678h, 1234h

.code
    mov eax, DWORD PTR wordList ; EAX = 12345678h
```

In the example above, the DWORD PTR operator is used to move two words from the wordList array into the lower and upper halves of the EAX register, effectively combining them into a doubleword.

The ALIGN directive and the PTR operator are important tools in assembly language programming to control memory alignment and ensure that data is properly interpreted based on its size when performing data transfers and operations.

==========

ALIGN directive

==========

The ALIGN directive is used to align a variable on a byte, word, doubleword, or paragraph boundary. The syntax is:

ALIGN bound where **bound is 1, 2, 4, or 16.**

The ALIGN directive is used to improve the performance of certain instructions. For example, the **FPU (floating-point unit)** can process data more quickly if it is aligned on a **doubleword boundary.**

The **ALIGN directive** and the **PTR operator** serve different purposes in assembly language programming, and they are not the same thing. Let's clarify the differences between them and provide examples of each:

The ALIGN directive is used to align a variable or label on a specific memory boundary. It ensures that the **memory address of the variable is a multiple of the specified alignment boundary.**

This can be useful for **optimizing memory access** and ensuring proper data alignment for certain data types, especially when working with **structured data** or **memory-mapped devices.**

Example of ALIGN directive:

```
.data
myData BYTE 10       ; Declare a byte-sized variable
ALIGN 4              ; Align the next variable on a 4-byte boundary
myDouble DWORD 20    ; Declare a doubleword-sized variable
```

In this example, the ALIGN directive is used to align myDouble on a 4-byte boundary. This ensures that myDouble starts at an address that is a multiple of 4.

==========

BYTE PTR Operator

==========

The **BYTE PTR operator** is used to access a single byte of a variable. The **PTR operator** is used in combination with standard assembler data types to specify the size of the data being accessed or manipulated. It is used when moving data between memory and registers to ensure that the correct number of bytes are read or written.

The following code shows how to use the BYTE PTR operator to access a single byte of the variable myDouble:

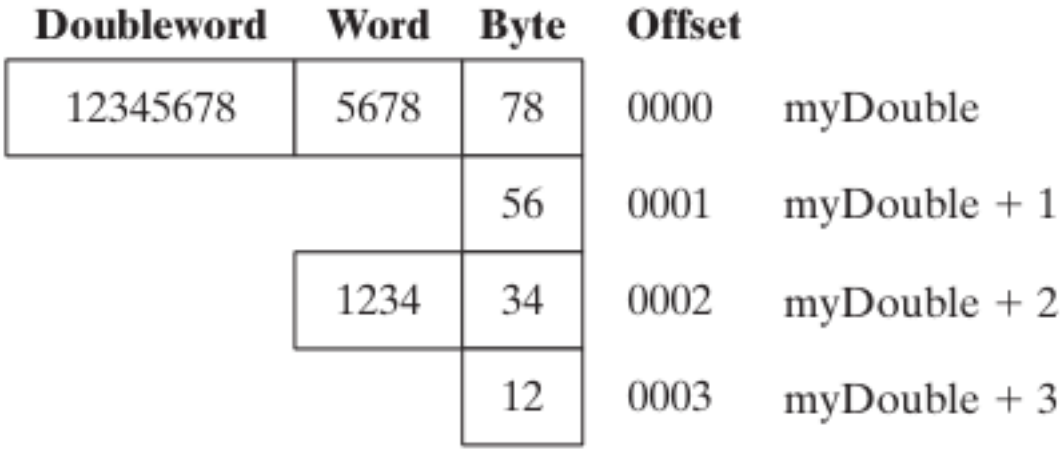
myDouble is a **doubleword(4 bytes)**, so the first byte is 78h.

```
.data
myDouble DWORD 12345678h

.code
mov bl, BYTE PTR myDouble ; BL = 78h
```

This code will move the value of the least significant byte of the variable myDouble into the register BL.

Memory Layout of the myDouble Variable. The following image shows the memory layout of the myDouble variable:



The least significant byte of the variable myDouble is at offset 0000h. The most significant byte of the variable myDouble is at offset 0003h.

Moving Smaller Values into Larger Destinations

We can use the BYTE PTR operator to move smaller values into larger destination operands. For example, the following code moves two words from memory to the register EAX:

```
.data
wordList WORD 5678h, 1234h

.code
mov eax, DWORD PTR wordList
```

This code will move the two words from the wordList array into the lower half and upper half of the register EAX, respectively.

```
.data
    myValue DWORD 12345678h ; Declare a doubleword-sized variable

.code
    mov eax, DWORD PTR myValue ; Load the entire DWORD into EAX
    mov al, BYTE PTR myValue   ; Load only the lowest byte into AL
```

NB: eax is 32 bit register.

In the first mov instruction, the PTR operator is used to specify that the entire DWORD stored in myValue should be loaded into the EAX register.

In the second mov instruction, the PTR operator is used to specify that only the lowest byte (LSB) of myValue should be loaded into the AL register.

Differences:

1. Purpose: ALIGN is used for memory alignment, ensuring that variables are properly positioned in memory for efficient access. PTR is used to specify the size of data when performing data transfers or operations.

2. Usage: ALIGN is a directive that appears in the data section to align variables. PTR is an operator used in instructions to specify the size of data during data manipulation.

In summary, ALIGN is used to control memory alignment, while PTR is used to specify data size in instructions. They serve different purposes and are used in different contexts in assembly language programming.