Performance of Aligned and Misaligned Structs

Aligned structure members can be accessed by the CPU more efficiently than misaligned structure members. This is because the CPU can perform fewer instructions to access aligned data.

The performance impact of misaligned structure members depends on the CPU architecture and the specific structure layout. However, in general, misaligned structure members can lead to a decrease in performance.

The following code shows a simple test to compare the performance of aligned and misaligned structure members:

```
1035 .data
1036     ALIGN DWORD
1037     startTime DWORD ?
1038     ; align startTime
1039     emp Employee <>
1040     ; or: emp EmployeeBad <>
1041 .code
1042     call
1043     GetMSeconds
1044     ; get starting time
1045     mov
1046     startTime,eax
1047     mov
1048     ecx,0FFFFFFFFh
1049     ; loop counter
1050     L1:
1051     mov
1052     emp.Years,5
1053     mov
1054     emp.SalaryHistory,35000
1055     loop
1056     L1
1057     call
1058     GetMSeconds
1059     ; get starting time
1060     sub
1061     eax,startTime
1062     call
1063     WriteDec
1064     ; display elapsed time
```

This code gets the system time, executes a loop that accesses structure fields, and calculates the elapsed time. The variable emp can be declared as an Employee or EmployeeBad object.

The Employee structure is aligned, while the EmployeeBad structure is not aligned.

not aligned.

When the code is executed using the Employee structure, the elapsed time is 6141 milliseconds.

When the code is executed using the EmployeeBad structure, the elapsed time is 6203 milliseconds.

The difference in elapsed time is small (62 milliseconds), but it is still measurable.

This suggests that even a small amount of misalignment can have a negative impact on performance.

It is important to note that the performance impact of misaligned structure members can vary depending on the CPU architecture and the specific structure layout.

In some cases, the performance impact may be more significant than the 62 milliseconds observed in the previous example.

Conclusion
It is generally a good practice to align structure members. This will help to ensure that the CPU can access the structure members efficiently, which can lead to improved performance.

Example: Displaying the System Time

The following is a step-by-step explanation of the program ShowTime.asm in depth:

```
; Structures (ShowTime.ASM)
INCLUDE Irvine32.inc
.data
    sysTime SYSTEMTIME <>
    XYPos COORD <10,5>
    consoleHandle DWORD ?
    colonStr BYTE ":",0
.code
    main PROC
    ; Get the standard output handle for the Win32 Console.
    INVOKE GetStdHandle, STD_OUTPUT_HANDLE
    mov consoleHandle,eax
```

The first part of the program defines the data structures that will be used.

The sysTime variable is a SYSTEMTIME structure, which will be used to store the system time.

The XYPos variable is a COORD structure, which will be used to store the cursor position.

The consoleHandle variable is a DWORD variable, which will be used to store the handle to the standard output handle.

The GetStdHandle function is used to retrieve the handle to the standard output handle.

The handle to the standard output handle is used by the SetConsoleCursorPosition and WriteString functions to write data to the console.

```
; Set the cursor position and get the system time.
INVOKE SetConsoleCursorPosition, consoleHandle, XYPos
INVOKE GetLocalTime, ADDR sysTime
```

The next part of the program sets the cursor position to the specified coordinates and retrieves the system time.

The SetConsoleCursorPosition function is used to set the cursor

position to the specified coordinates. The GetLocalTime function is used to retrieve the system time.

```
1090 ; Display the system time (hh:mm:ss).
1091 movzx eax,sysTime.wHour
1092 ; hours
1093 call WriteDec
1094 mov edx,OFFSET colonStr
1095 ; ":"
1096 call WriteString
1097 movzx eax,sysTime.wMinute
1098 ; minutes
1099 call WriteDec
1100 call WriteString
1101 movzx eax,sysTime.wSecond
1102 ; seconds
1103 call WriteDec
1104 call Crlf
```

The next part of the program displays the system time to the console.

The WriteDec function is used to write a decimal number to the console.

The WriteString function is used to write a string to the console.

The Crlf function is used to write a carriage return and line feed to the console.

```
1108 call WaitMsg
1109 ; "Press any key..."
1110 exit
1111 main ENDP
1112 END main
```

The last part of the program displays a message to the console and waits for the user to press a key.

The WaitMsg function is used to display a message to the console and wait for the user to press a key.

Conclusion

The ShowTime.asm program is a simple example of how to use structures in assembly language. By using structures, you can group related data together and make your code more readable and maintainable.