Mixed-Mode Arithmetic

The passage below describes mixed-mode arithmetic in assembly language.

Mixed-mode arithmetic is arithmetic involving both integers and reals.

The Intel instruction set provides instructions that promote integers to reals and load the values onto the floating-point stack.

To perform mixed-mode arithmetic in assembly language, you must first use an instruction to promote the integer to a real.

For example, the following instruction promotes the integer in the N register to a real and loads it onto the floating-point stack:

```
fild N
```

Once you have finished performing arithmetic operations on the real, you can use an instruction to store it back to memory as an integer.

For example, the following instruction stores the real in the ST(0) register to the Z integer variable:

```
fist Z
```

It is important to note that the FIST instruction rounds the real to the nearest integer. If you need to truncate the real to an integer, you can use the following code:

```
fstcw ctrlWord     ; Store the FPU control word in ctrlWord
or ctrlWord, 110000000000b ; Set RC (Rounding Control) to truncate
fldcw ctrlWord     ; Load the modified control word to change the rounding mode

; Perform arithmetic operations here using the modified rounding mode
fild N             ; Load the integer N into ST(0)
fadd X             ; Add the real X to ST(0)
fist Z             ; Store the truncated result in integer Z

fstcw ctrlWord     ; Store the FPU control word in ctrlWord again
and ctrlWord, 001111111111b ; Reset the rounding mode to default (round to nearest)
fldcw ctrlWord     ; Load the control word to restore the default rounding mode
```

In this code, we first store the FPU control word in the variable ctrlWord.

Then, we modify the **RC field** in ctrlWord to set the **rounding mode to truncate.**

After changing the **rounding mode**, we perform the desired arithmetic operations.

After completing the arithmetic operations, we store the FPU control word again and reset the rounding mode to the default setting (round to nearest) by using a bitwise AND operation to clear the bits that control the rounding mode.

Finally, we load the modified control word to ensure the default rounding mode is restored.

Example 1: Adding an Integer to a Double

In this example, you're adding an integer (N) to a double (X) and storing the result in a double (Z).

The C++ code automatically promotes the integer to a real before performing the addition. The equivalent assembly code for this operation is as follows:

```
.data
    N SDWORD 20
    X REAL8 3.5
    Z REAL8 ?

.code
    fild N    ; Load the integer into ST(0)
    fadd X    ; Add the memory value (X) to ST(0)
    fstp Z    ; Store ST(0) to memory (Z)
```

This code loads the integer N into ST(0), adds the real value X, and stores the result in the real Z.

Example 2: Promoting and Truncating

In this example, you're promoting an integer (N) to a double and evaluating a real expression involving N and X.

The result is stored in an integer (Z). C++ typically performs the conversion automatically, but in assembly, you use FIST to convert the result to an integer. Here's the code:

```
fild N    ; Load the integer into ST(0)
fadd X    ; Add the real X to ST(0)
fist Z    ; Store ST(0) to the integer Z
```

This code loads the integer N into ST(0), adds the real X, and then stores the truncated result in the integer Z.

Changing the Rounding Mode:

The FPU control word's RC field allows you to specify the rounding mode. You can use the **FSTCW instruction to store the control word in a variable, modify the RC field** to change the rounding mode (e.g., truncate or round to nearest), and **then use the FLDCW instruction to load the modified control word back into the FPU.** This allows you to control how rounding is performed during floating-point operations.

Example 3:

```
620 INCLUDE Irvine32.inc
621 .data
622     N           SDWORD 20       ; Integer value
623     X           REAL8 3.5        ; Double-precision real value
624     Z           REAL8 ?         ; Result stored as a double-precision real
625     ctrlWord    WORD 0          ; FPU control word
626
627 .code
628     main PROC
629         ; Initialize FPU
630         finit
631
632         ; Store the FPU control word
633         fstcw ctrlWord
634         ; Modify the control word to set the rounding mode to truncate
635         or ctrlWord, 110000000000b
636         fldcw ctrlWord
637         ; Perform mixed-mode arithmetic
638         fild N          ; Load integer N into ST(0)
639         fadd X          ; Add real X to ST(0)
640         fstp Z          ; Store the result in Z
641         ; Display the result
642         mov edx, OFFSET Z
643         call WriteFloat
644         ; Reset the rounding mode to default (round to nearest)
645         and ctrlWord, 001111111111b
646         fldcw ctrlWord
647         exit
648     main ENDP
649 END main
```

Initialization:

INCLUDE Irvine32.inc: This includes the Irvine32 library, which provides various I/O and utility functions for assembly language programming.

This section is used for defining data variables.

N SDWORD 20: Declares a signed doubleword (32-bit) integer variable named N with an initial value of 20.

X REAL8 3.5: Declares an 8-byte (64-bit) double-precision real variable named X with an initial value of 3.5.

Z REAL8 ?: Declares another double-precision real variable named Z

without an initial value. It will store the result of the computation.

ctrlWord WORD 0: Declares a 16-bit variable named ctrlWord to store the FPU control word.

FPU Initialization:

finit: Initializes the FPU (Floating-Point Unit), ensuring a clean start for FPU operations.
Storing the FPU Control Word:

fstcw ctrlWord: Stores the current FPU control word in the variable ctrlWord. This allows us to modify the control word without affecting the system's default FPU settings.
Modifying the Rounding Mode:

or ctrlWord, 110000000000b: This binary OR operation modifies bits 10 and 11 of the control word to set the rounding control (RC) to truncate. Truncate mode discards the fractional part when converting real numbers to integers.

Mixed-Mode Arithmetic:

fild N: Loads the integer N into the FPU stack, promoting it to a double-precision real value in ST(0).

fadd X: Adds the real X to the value in ST(0), resulting in a mixed-mode addition. The result remains in ST(0).

fstp Z: Stores the result from ST(0) into the double-precision real variable Z.

Displaying the Result:

mov edx, OFFSET Z: Prepares the address of the Z variable for displaying the result.

call WriteFloat: Calls the WriteFloat procedure to display the result stored in Z.

Resetting the Rounding Mode:

and ctrlWord, 001111111111b: This binary AND operation resets bits 10 and 11 of the control word to their default values, effectively setting the rounding mode back to round to nearest even.

Program Exit:

exit: Exits the program.

In summary, this program demonstrates the following concepts:

Changing the FPU control word to set the rounding mode for FPU operations.

Performing mixed-mode arithmetic by adding an integer and a double-precision real value using FPU instructions.

Controlling the rounding mode to ensure the desired behavior for the arithmetic operations.

The program calculates the result of mixed-mode arithmetic and displays it, considering the specified rounding mode, and then resets the rounding mode to its default state before exiting.