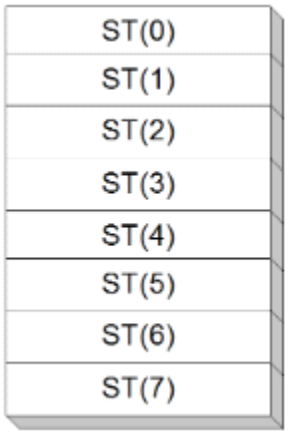
Floating Point Unit (FPU)

Muhammad Ahsan(P17-6142)

***Introduction*:**

A floating point unit (FPU) is a specialized processor that manipulates numbers more quickly than the basic microprocessor. The FPU does this by means of instructions that focus entirely on large mathematical operations. In the early history of Intel, none of their processors had a built-in floating-point capability. If you wanted floating-point processing, you emulated it with software. If REAL4's accuracy and precision are not acceptable, use REAL8 instead. In C, it was decided that REAL4 is float and REAL8 is double.



The FPU simply has eight identical 80-bit registers. The 80-bit registers are generally designated as a stack of eight registers. If a stack is instructed to load a value, the value is loaded into ST(7), if another value is needed to be loaded in, then the ST(6) register is used. This operates on (LIFO) Last In First Out, so the top stack is always affected. So in FPU, they would always be inserted into ST(0), the first value pushed would be in the ST(0) register, if another value is pushed, then the first pushed value would be ST(1). The newest push would always be on top and on ST(0).

Data Types:

In the context of FPU operations, integers are whole numbers. All integers used in FPU instructions are also considered as signed integers, the most significant bit being 0 for positive values or 1 for negative values. The three integer data types are: the 16-bit WORD, the 32-bit DWORD, and the 64-bit QWORD. The floating point data types are simply binary numbers represented in a manner similar to the scientific notation used for decimal values.
For example: 322 = 3.22 x = (3.22E+0002) Within the floating point data types, three sizes of real numbers are available: 32-bit REAL4 (short real or single precision), 64-bit REAL8 (long real or

double precision), 80-bit REAL10 (temporary real or extended precision). There is another third data type which is more or less a data type, NAN’s (Not A Number). This is for indefinite or infinite numbers. For this data type, all the exponents are set to 1.

Format:

Looking at a 32 bit float, the floating point number is divided into three sections: Sign (1 bit), Exponent (8 bit) and Mantissa (23 bit).
Here the sign bit represents whether the number is positive (0) or negative (1), the exponent represents
the power of 2 obtained after shifting the decimal point then biased by adding to 127 while the mantissa
represents the numbers after the decimal point. For example:
45 = 101101.01 //Binary representation with 1100 repeating.
45 = 1.0110101 X 2 5 //Shifting decimal while increasing the power of 2
Sign = 0 //Positive value
Exponent = 5 + bias = 132 = 1000 0100
Mantissa = 0110101

Basic Arithmetic Instructions (Floating-Point):

FABS	Absolute value of ST(0)
FADD	Add two floating point values
FADDP	Add two floating point values and Pop ST(0)
FCHS	Change the Sign of ST(0)
FDIV	Divide two floating point values
FDIVP	Divide two floating point values and Pop ST(0)
FDIVR	Divide in Reverse two floating point values
FDIVRP	Divide in Reverse two floating point values and Pop ST(0)
FMUL	Multiply two floating point values
FMULP	Multiply two floating point values and Pop ST(0)
FRNDINT	Round ST(0) to an Integer
FSQRT	Square Root of ST(0)
FSUB	Subtract two floating point values
FSUBP	Subtract two floating point values and Pop ST(0)
FSUBR	Subtract in Reverse two floating point values
FSUBRP	Subtract in Reverse two floating point values and Pop ST(0)

Data Transfer Integer number:

FILD	Integer Load from memory
FIST	Integer Store to memory
FISTP	Integer Store to memory and Pop the top data register

Data Transfer Real number:

FCMOVcc	Conditional Move based on CPU flags
FLD	Load real number from memory
FST	Store real number to memory
FSTP	Store real number to memory and Pop the top data register
FXCH	exchange the top data register with another data register
FLD1	Load the value of 1
FLDL2E	Load the Log base 2 of e
FLDL2T	Load the Log base 2 of ten
FLDLG2	Load the Log base 10 of 2 (common log of 2)
FLDLN2	Load the Log base *e* of 2 (natural log of 2)
FLDPI	Load the value of PI
FLDZ	Load the value of Zero

Logrithmic Functions of FPU:

F2XM1	2 to the X power Minus 1
FSCALE	SCALE ST(0) by ST(1)
FYL2X	$ST(1)*Log_3(ST(0))$
FYL2XP1	$ST(1)*Log_2(ST(0)+1)$