FLOATING POINT ALU

While performing arithmetic operations, a Floating point number is treated as two fixed point numbers: Exponent and Mantissa.

Consider X and Y as two Floating point numbers.

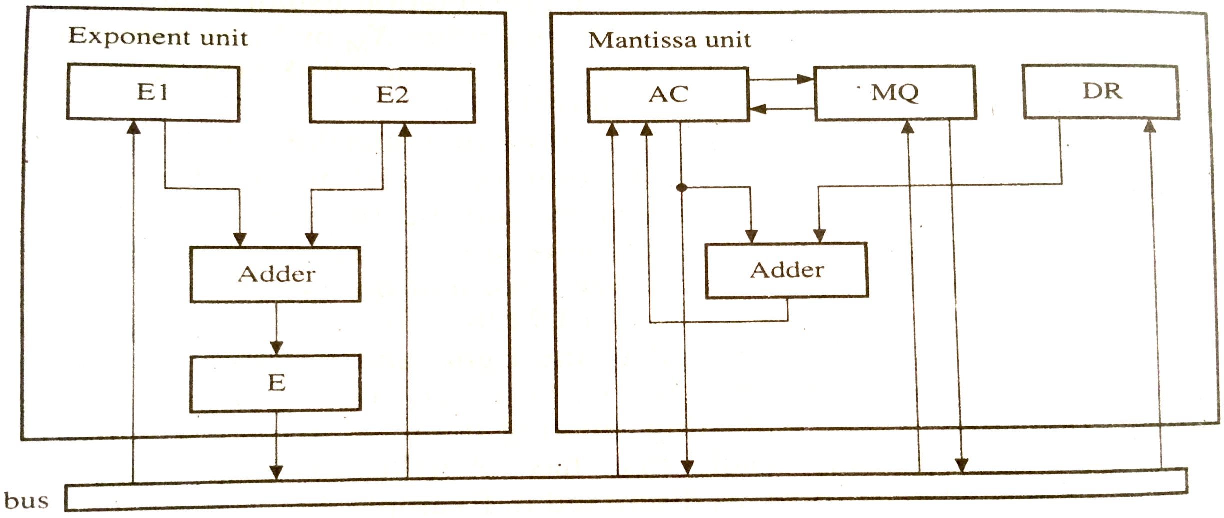
$X = X_M . 2^{X_E}$

$Y = Y_M . 2^{Y_E}$

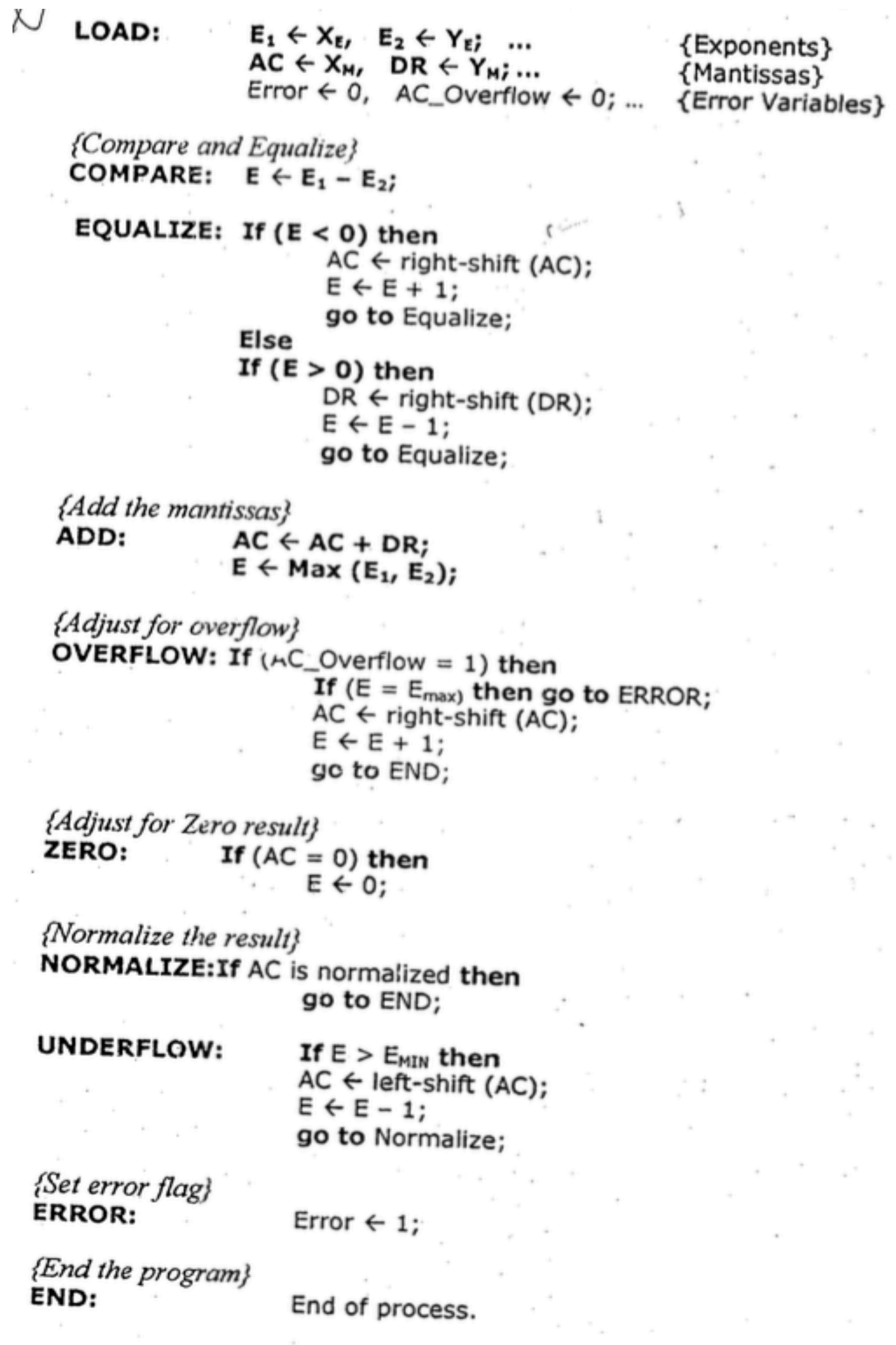
Then the arithmetic operations on these two numbers will be performed as:

OPERATION	MANTISSA	EXPONENT
Addition	Add	Equalize
Subtraction	Sub	Equalize
Multiplication	Mul	Add
Division	Div	Sub

From the above table it is evident that Mantissa and Exponent are dealt with, in different ways. Hence a Floating Point ALU has two units internally: Exponent Unit and Mantissa Unit.



FLOATING POINT ADDITION ALGORITHM



```
LOAD:        $E_1 \leftarrow X_E$,   $E_2 \leftarrow Y_E$;  ...          {Exponents}
             AC ← $X_M$,   DR ← $Y_M$; ...                   {Mantissas}
             Error ← 0,   AC_Overflow ← 0; ...            {Error Variables}

{Compare and Equalize}
COMPARE:     E ← $E_1 - E_2$;

EQUALIZE:    If (E < 0) then
                   AC ← right-shift (AC);
                   E ← E + 1;
                   go to Equalize;
             Else
             If (E > 0) then
                   DR ← right-shift (DR);
                   E ← E – 1;
                   go to Equalize;

{Add the mantissas}
ADD:         AC ← AC + DR;
             E ← Max ($E_1$, $E_2$);

{Adjust for overflow}
OVERFLOW: If (AC_Overflow = 1) then
                   If (E = $E_{max}$) then go to ERROR;
                   AC ← right-shift (AC);
                   E ← E + 1;
                   go to END;

{Adjust for Zero result}
ZERO:        If (AC = 0) then
                   E ← 0;

{Normalize the result}
NORMALIZE:If AC is normalized then
                   go to END;

UNDERFLOW:         If E > $E_{MIN}$ then
                   AC ← left-shift (AC);
                   E ← E – 1;
                   go to Normalize;

{Set error flag}
ERROR:             Error ← 1;

{End the program}
END:               End of process.
```