fsqrt rationale (or at least, some sense for the magic constant)

I. J. Mermet

December 11, 2016

1 Explicacion de la obtencion de la constante

Sea f un numero flotante de precision simple conforme norma IEEE 754.

$$f = (-1)^s.(1 + M).2^{exp} \tag{1}$$

Queremos encontrar una forma de calcular $\sqrt{f}$, por lo tanto asumiremos que s vale 0.

Sea $y = \sqrt{f} = f^{1/2}$, entonces:

$$\log_2(y) = \frac{1}{2}\log_2(f) \tag{2}$$

La ecuacion (2) nos interesa en particular, pues es facil de calcular:

$$f = (1 + M) * 2^{exp} \tag{3}$$

$$\log_2(f) = \log_2((1 + M) * 2^{exp}) = exp + \log_2(1 + M) \tag{4}$$

Conforme IEEE 754, $M \in [\,0, 1) \Rightarrow (1 + M) \in [\,1, 2) \Rightarrow \log_2(1 + M) \in [\,0, 1) \Rightarrow \log_2(1 + M) \simeq M \Rightarrow \log_2(1 + M) = M + \sigma$

Tomando en cuenta el anterior resultado, y la ecuacion (4), reemplazando:

$$\log_2(f) = exp + M + \sigma \tag{5}$$

Sea I_f la representacion del mapa de bits de f como un numero entero. Entonces, tenemos que:

$$I_f = E_f * L + M_f \tag{6}$$

Donde

- $E_f = exp + B$
- $B =$ bias del exponente (127)
- $M_f = M * L$

- $L = 2^{23}$ (23 es la cantidad de bits de la mantisa)

$I_f = (exp{+}B){*}L{+}M{*}L = L{*}(\,exp{+}B{+}M) = L{*}(\,exp{+}B{+}M{+}\sigma{-}\sigma) = L * (\,\log_2(f) + B - \sigma) \Rightarrow \frac{I_f}{L} - (\,B - \sigma) = \log_2(f)$

Volviendo a (2): $\frac{I_y}{L} - (\,B - \sigma) = \frac{1}{2}(\,\frac{I_f}{L} - (\,B - \sigma)\,) \Rightarrow \frac{I_y}{L} = \frac{1}{2}(\,\frac{I_f}{L} - (\,B - \sigma)\,) + (\,B - \sigma) \Rightarrow \frac{I_y}{L} = \frac{I_f}{2L} + \frac{(\,B - \sigma)}{2} \Rightarrow I_y = \frac{I_f}{2} + \frac{(\,B - \sigma) * L}{2}$

Tomando $\sigma = 0$, para simplificar el razonamiento, el termino $\frac{B*L}{2}$ es `0x3f800000`. Convirtiendo I_y a una interpretacion de su mapa de bits como flotante de precision simple, tenemos una aproximacion del valor buscado. Su precision puede mejorarse buscando un σ que minimice el error medio para todos los puntos o aplicando `n` iteraciones de Newton-Raphson.

2 Codigo inicial

```
float fsqrt(float n) {
  unsigned i = *(unsigned*)&n;
  i = (i + 0x3f800000) >> 1;
  float y = *(float*)&i;
  return y;
}
```

3 Aplicando iteraciones de Newton-Raphson

La funcion raiz es derivable en el intervalo de representacion de flotantes no negativos, por lo que usaremos iteraciones de Newton-Raphson para mejorar los resultados obtenidos anteriormente. Usaremos el valor aproximado previamente como raiz para el metodo.

Se define que:

$$X_{n+1} = X_n - \frac{g(X_n)}{g'(X_n)} \tag{7}$$

Definimos

$$g(y) = y^2 - f \tag{8}$$

Donde f es un numero real. $g(y) = 0$ si $y = \sqrt{f}$. Derivamos (8) para obtener

$$g'(y) = 2y \tag{9}$$

$$y_{n+1} = y_n - \frac{y_n^2 - f}{2y_n} = y_n - (\,\frac{y_n}{2} - \frac{f}{2y_n}) = \frac{y_n}{2} + \frac{f}{2y_n} \tag{10}$$

Implementamos una iteracion de Newton-Raphson:

```
float fsqrt(float n) {
  unsigned i = *(unsigned*)&n;
  i = (i + 0x3f800000) >> 1;
  float y = *(float*)&i;
  y = y*0.5f + n/(2*y);
  return y;
}
```

Se pueden incluir multiples iteraciones de Newton-Raphson en pos de mejorar la precision, pero con una penalidad de tiempo.

4 Consideraciones a futuro

A fin de mejorar la precision del algoritmo sin afectar el tiempo, se deberia buscar un σ que como se menciono previamente, minimice el error medio para todos los puntos.