Московский государственный технический университет им.Н.Э.Баумана
Калужский филиал
Приборостроительный факультет
Кафедра П2-КФ

Отчет

по лабораторной работе №1

«Исследование особенностей реализации арифметических операций на языке ассемблера»

по курсу

«Теория и проектирование ЭВМ и ВС»

Студент : Комиссаров А.В., гр.ЭВМ-92

Преподаватель : доцент Максимов А.В.

Калуга, 1998 г.

Содержание.

1. Задание на лабораторную работу

Вариант по списку - **№25** :

X_{max}	Y_{max}	A	B	N*
234	16	34	14,2	32; 8; 16

Требуется выполнить масштабирование следующих арифметических операций :

- сложение (Z = X + Y);
- вычитание (Z = X – Y);
- умножение (Z = X * Y);
- умножение на постоянное число (Z = A * X);
- деление (Z = X / Y);
- деление постоянной на переменную (Z = A / X);
- деление переменной на постоянную (Z = X / B).

Одна из операций (умножение) дополнительно масштабируется в предельных масштабах.

Написать и отладить программу выполнения операций на языке ассемблера. Исследовать особенности выполнения операций на тестовых вариантах (предложенных студентом) и проверить работоспособность программ на вариантах, указанных преподавателем. Кроме того, для операции умножения опеределить запас по точности, реализовать программу повышенной точности умножения, оценить количественно повышение точности, построить график зоны нулевого машинного результата. Для операций деления с постоянной также выполнить программу повышения точности, сопоставить результаты с делением по общим правилам.

2. Расчет масштабных коэффициентов

2.1. Масштабные коэффициенты операндов (двоичные масштабы)

Выбираем дробную арифметику (двоичные масштабы); в этом случае будут справедливы следующие соотношения для масштабных коэффициентов:

$$\mathcal{M}_X = \frac{1-2^{-n_X}}{2^{m_X}} \approx 2^{-m_X} \quad , \qquad (1)$$

где $\mathcal{M}_X$ - масштабный коэффициент числа *X*;

n_X - число битов, отводимых под мантиссу разрядной сетки операнда;

m_X - число, определяемое из соотношения : $2^{m_X - 1} \le \left|X_{max}\right| < 2^{m_X}$.

---

* Соответственно указаны количества бит разрядной сетки операндов при выполнении операции сложения / вычитания; умножения; деления.

В нашем случае имеем :

$$2^4 = 16 < 2^5, \text{ поэтому } m_Y = 5; \text{ следовательно, } \mathcal{M}_Y = 2^{-5};$$
$$2^7 < 234 < 2^8, \text{ поэтому } m_X = 8; \text{ следовательно, } \mathcal{M}_X = 2^{-8};$$
$$2^5 < 34 < 2^6, \text{ поэтому } m_A = 6; \text{ следовательно, } \mathcal{M}_A = 2^{-6};$$
$$2^3 < 14{,}2 < 2^4, \text{ поэтому } m_B = 4; \text{ следовательно, } \mathcal{M}_B = 2^{-4}.$$

Учитывая, что в приближенном равенстве (1) двоичный масштабный коэффициент не зависит от числа разрядов, полученные результаты для масштабных коэффициентов исходных операндов будут справедливы для всех операций в данной лабораторной работе (в случае целочисленной арифметики это утверждение было бы неверно).

2.2. Масштабные коэффициенты результата операции сложения / вычитания

- *Без повышенной точности результата.*

$$\mathcal{M}_Z = 2^{-1} \min\{\mathcal{M}_X, \mathcal{M}_Y\} = 2^{-1} \min\{2^{-8}, 2^{-5}\} = 2^{-9}.$$

- *С повышенной точностью результата.*

$Z_{\max} = X_{\max} + Y_{\max}$, где $Z_{\max}$ - наибольшее значение результата, которое возможно при заданных максимальных значениях операндов; $Z_{\max} = 250$; масштабный коэффициент, соответствующий максимально возможному результату : $\mathcal{M}_Z^* = 2^{-8}$, следовательно,

$$\mathcal{M}_Z = \min\{\mathcal{M}_X, \mathcal{M}_Y, \mathcal{M}_Z^*\} = \min\{2^{-8}; 2^{-5}; 2^{-8}\} = 2^{-8}.$$

2.3. Масштабные коэффициенты результата операции умножения

- *Без повышенной точности результата* : учитывая, что микропроцессоры фирмы INTEL обрабатывают информацию младшими разрядами вперед, применим следующее выражение :

$$\mathcal{M}_Z = 2^{-1} \mathcal{M}_X \mathcal{M}_Y = 2^{-14};$$

- *С повышенной точностью результата* : в этом случае масштабный коэффициент совпадет с масштабным коэффициентом, рассчитанным исходя из максимально возможного результата, получаемого в процессе выполнения операции умножения :

$$Z_{\max} = X_{\max} Y_{\max} = 234 \cdot 16 = 3744;$$
$$2048 = 2^{11} < 3744 < 2^{12} = 4096,$$

следовательно,

$$\mathcal{M}_Z^* = 2^{-12}.$$

2.4. Масштабные коэффициенты результата операции умножения переменной *X* на константу *A*

2.4.1. Первый способ. Если принять машинное значение результата равным машинному значению операнда-переменной, т.е. $\overline{Z} = \overline{X}$, то получим недвоичный масштабный коэффициент :

$$\mathcal{M}_Z = \frac{\mathcal{M}_X}{A} = \frac{2^{-8}}{2^5 + 2^1} = \frac{1}{2^{13} + 2^9}. \qquad (2)$$

2.4.2. Второй способ (обычный). В этом случае масштабный коэффициент вычисляется обычным способом (см. п.2.3) :

$$\mathcal{M}_Z = 2^{-1} \mathcal{M}_A \mathcal{M}_X = 2^{-1} 2^{-8} 2^{-6} = 2^{-15} \text{ (без повышения точности);}$$

В случае повышенной точности представления результата имеем :

$$Z_{\max} = A\, X_{\max} = 34 \cdot 234 = 7956;$$

$$4096 = 2^{12} < 7956 < 2^{13} = 8192,$$

следовательно,

$$\mathcal{M}_Z = 2^{-13}.$$

2.4.3. Третий способ (способ сдвига и сложения). В данном случае, вообще говоря, масштабный коэффициент определяется простым произведением масштабных коэффициентов операндов *A* и *X*, так как операция реализуется на программном уровне. Но в нашем случае появляется возможность увеличить масштабный коэффициент результата на 2 за счет отсутствия «лишнего» сдвига (при отсутствии переполнения результата) каждого из слагаемых результата вправо (об этом более подробно будет сказано ниже, при анализе соответствующей операции). Следовательно,

$$\mathcal{M}_Z = 2^1 \mathcal{M}_A \mathcal{M}_X = 2^1 2^{-8} 2^{-6} = 2^{-13}. \qquad (3)$$

2.5.Масштабный коэффициент результата операции деления

В случае операции деления масштабный коэффициент результата может быть вычислен следующим образом :

$$\mathcal{M}_Z = 2^{-n} \cdot \frac{\mathcal{M}_X}{\mathcal{M}_Y} = 2^{-15} \cdot \frac{2^{-8}}{2^{-5}} = 2^{-18}.$$

2.6.Масштабный коэффициент результата операции деления постоянной *A* на переменную *X*

В данном случае масштабный коэффициент определяется аналогично случаю обычного деления переменной на переменную (см. п.2.4) :

$$\mathcal{M}_Z = 2^{-n} \cdot \frac{\mathcal{M}_A}{\mathcal{M}_X} = 2^{-15} \cdot \frac{2^{-6}}{2^{-8}} = 2^{-13}.$$

2.7. Масштабные коэффициенты результата операции деления переменной *X* на постоянную *B*

- *Без повышенной точности результата.* В данном случае вычисление масштабного коэффициента результата осуществляется обычным способом :

$$\mathcal{M}_Z = 2^{-n} \cdot \frac{\mathcal{M}_X}{\mathcal{M}_B} = 2^{-15} \cdot \frac{2^{-8}}{2^{-4}} = 2^{-19}.$$

- *С повышенной точностью результата.* В данном случае масштаб частного определяется по общим правилам масштабирования, исходя из максимального значения результата :

$$Z_{\max} = \frac{X_{\max}}{B} \approx 16{,}478873;$$

$$16 = 2^4 < 16{,}478873 < 2^5 = 32,$$

следовательно,

$$\mathcal{M}_Z^* = 2^{-5}.$$

2.7. Масштабные коэффициенты операндов и результата операции умножения (случай предельных масштабов)

Предельные масштабные коэффициенты операндов могут быть определены следующим образом :

$$\mathcal{M}_X = \frac{\overline{X}_{\max}}{|X_{\max}|} \cong \frac{1}{2^7 + 2^6 + 2^5 + 2^3 + 2^1} \approx 0{,}00424011;$$

$$\mathcal{M}_Y = \frac{\overline{Y}_{\max}}{|Y_{\max}|} \cong \frac{1}{2^4} = 2^{-4} \approx 0{,}0620117;$$

$$\mathcal{M}_Z = 2^{-1}\, \mathcal{M}_X \mathcal{M}_Y =$$

$$= \frac{2^{-5}}{2^7 + 2^6 + 2^5 + 2^3 + 2^1} = \frac{1}{2^{12} + 2^{11} + 2^{10} + 2^8 + 2^6} \approx 0{,}000133547.$$

3. Алгоритмы выполнения арифметических операций и фрагменты программы на языке ассемблера. Экспериментальный анализ операций

3.1. Операция сложения / вычитания

Коэффициенты выравнивания масштабов :

- *Без повышения точности* (см. п. 2.2):

$$K_X = \frac{\mathcal{M}_Z}{\mathcal{M}_X} = \frac{2^{-9}}{2^{-8}} = 2^{-1}; \;\; K_Y = \frac{\mathcal{M}_Z}{\mathcal{M}_Y} = \frac{2^{-9}}{2^{-5}} = 2^{-4}.$$

- *С повышением точности* (изменится масштабный коэффициент результата, см. п. 2.2):

$$K_X = \frac{\mathcal{M}_Z}{\mathcal{M}_X} = \frac{2^{-8}}{2^{-8}} = 2^{0}; \;\; K_Y = \frac{\mathcal{M}_Z}{\mathcal{M}_Y} = \frac{2^{-8}}{2^{-5}} = 2^{-3}$$

3.1.1. Алгоритм выполнения операции :

- Выравниваем первое слагаемое* : $\overline{X}_1 = \overline{X} \cdot K_X$; $\overline{X}_1 = \overline{X} \cdot 2^{-1}$ (без повышения точности, т.е. первый вариант); $\overline{X}_1 = \overline{X} \cdot 2^{0} \equiv \overline{X}$ (с повышением точности, т.е. второй вариант);
- Выравниваем второе слагаемое : $\overline{Y}_1 = \overline{Y} \cdot K_Y$; $\overline{Y}_1 = \overline{Y} \cdot 2^{-4}$ (без повышения точности, т.е. первый вариант); $\overline{Y}_1 = \overline{Y} \cdot 2^{-3}$ (с повышением точности, т.е. второй вариант);
- Вычисляем машинный результат : $\overline{Z} = \overline{X}_1 \pm \overline{Y}_1$.

Заметим, что машинные представления операндов могут быть вычислены по формулам : $\overline{X} = X \cdot \mathcal{M}_X$; $\overline{Y} = Y \cdot \mathcal{M}_Y$. Соответственно $Z = \dfrac{\overline{Z}}{\mathcal{M}_Z}$.

3.1.2. Фрагмент программы на языке ассемблера, реализующий алгоритм сложения / вычитания машинных представлений чисел (без повышения точности результата)

```
mov   eax, X    ;Запись в регистр машинного представления первого операнда
mov   ebx, Y    ;Запись в регистр машинного представления второго операнда
sar   eax, 1    ;Выравнивание мантиссы первого операнда
sar   ebx, 4    ;Выравнивание мантиссы второго операнда
add   eax, ebx  ;Двоичное сложение выравненных мантисс
mov   Z, eax    ;Запись результата в ячейку основной памяти
...
```

* Умножение двоичного числа на степень числа 2 эквивалентно сдвигу этого двоичного числа на число разрядов, равное абсолютному значению показателя степени. При этом сдвиг осуществляется влево, если показатель степени положительный, и вправо, если показатель степени отрицательный.

3.1.3. Фрагмент программы на языке ассемблера, реализующий алгоритм сложения / вычитания машинных представлений чисел (без повышения точности результата)

```
mov   eax, X    ;Запись в регистр машинного представления первого операнда
mov   ebx, Y    ;Запись в регистр машинного представления второго операнда
sar   ebx, 3    ;Выравнивание мантиссы второго операнда
add   eax, ebx  ;Двоичное сложение выравненных мантисс
mov   Z, eax    ;Запись результата в ячейку основной памяти
…
```

3.1.4. Экспериментальный анализ операции сложения / вычитания

Для трассировки (пошагового прогона) программы была использована программа-отладчик Turbo Debugger фирмы Borland International (новое название фирмы – Inprise Corporation). Текущие значения регистров можно было просмотреть в отдельном окне.

- *Пример 1*. Возьмем максимальные значения операндов.

$$\overline{X}=\overline{X}_{\max} = X_{\max} \cdot \mathcal{M}_X =$$
$$= \left(2^7+2^6+2^5+2^3+2^1\right)\cdot 2^{-8} = 2^{-1}+2^{-2}+2^{-3}+2^{-5}+2^{-7};$$
$$\overline{Y}=\overline{Y}_{\max} = Y_{\max} \cdot \mathcal{M}_Y = 2^4 \cdot 2^{-5} = 2^{-1}.$$

Разрядные сетки машинных представлений исходных операндов, промежуточных значений и результата будут выглядеть следующим образом (значком «тильда» помечены знаковые биты)* :

$$\overline{X} = \quad \left[\tilde{0}111\ 0101\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 7500\ 0000\ h;$$
$$\overline{Y} = \quad \left[\tilde{0}100\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 4000\ 0000\ h;$$
$$\overline{X}\cdot 2^{-1} = \quad \left[\tilde{0}011\ 1010\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 3A80\ 0000\ h$$
$$\overline{Y}\cdot 2^{-4} = \quad \left[\tilde{0}000\ 0100\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 0400\ 0000 h;$$
$$\overline{X}\cdot 2^{-1}+\overline{Y}\cdot 2^{-4} = \quad \left[\tilde{0}011\ 1110\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 3E80\ 0000 h.$$

– –

$$\overline{Y}\cdot 2^{-3} = \quad \left[\tilde{0}000\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 0800\ 0000 h;$$
$$\overline{X}+\overline{Y}\cdot 2^{-3} = \quad \left[\tilde{0}111\ 1101\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\right] = 7D00\ 0000 h.$$

После горизонтальной черты приведен результат вычисления для случая с повышенной точностью**. Из анализа операции на данном примере можно заключить, что, во-первых, алгоритм успешно справился с максимальным результатом (переполнения разрядной сетки в результате вычислений не произошло). Физический результат, вычисленный как отношение машинного

---

* Необходимо учесть, что в операции сдвига участвует не все машинное представление числа целиком, а лишь его мантисса; знаковый бит при этом остается без изменения.
** Хотя в рассматриваемом примере и в первом случае был получен результат с максимальной точностью, т.к. разрядная сетка довольно большая.

результата к масштабному коэффициенту результата, совпадает с суммой физических величин исходных операндов, в чем нетрудно убедиться :

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-8}}{2^{-9}} =$$
$$= 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1 = 250 = 234 + 16.$$

Во-вторых, из сравнения результатов видно, что во втором случае удалось задействовать не использовавшийся ранее самый старший разряд мантиссы (вес 2^{-1}). При этом также получаем верный результат (с учетом удвоенного масштабного коэффициента результата) :

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-7}}{2^{-8}} =$$
$$= 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1 = 250 = 234 + 16.$$

Кроме того, в данном случае удалось сэкономить время выполнения операции, так как отсутствует строка *sar eax, 1* , а второй операнд сдвигается на 1 разряд меньше.

- *Пример 2*. Возьмем следующие физические значения исходных операндов :

$X = X_{\max} = 234$; $Y = Y_{\min} = -16$.

$$\overline{X} = \overline{X}_{\max} = X_{\max} \cdot \mathcal{M}_X =$$
$$= \left(2^7 + 2^6 + 2^5 + 2^3 + 2^1\right) \cdot 2^{-8} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7};$$

$$\overline{Y} = Y \cdot \mathcal{M}_Y = 2^4 \cdot 2^{-5} = 2^{-1}\,(\text{прямой код});\; \overline{Y} = \left[\tilde{1}011\,\underbrace{1111\ ...\ 1111}_{7\ \text{тетрад}}\right] (\text{обратный код});$$

$$\overline{Y} = \left[\tilde{1}100\ \underbrace{0000\ ...\ 0000}_{7\ \text{тетрад}}\right] = 2^{-1}\,(\text{дополнительный код})$$

Таким образом, с учетом знака исходные операнды, подлежащие занесению в ЭВМ : 7500 0000 h и C000 0000 h.

Осуществив прогон программы с указанными машинными представлениями операндов, получим машинные результаты : 3680 0000 h = $\left[\tilde{0}011\,0110\,1000\,0000 \ldots 0000\right]$ и 6D00 0000 h = = $\left[\tilde{0}110\,1101\,0000\,0000 \ldots 0000\right]$ (соответственно по алгоритму без повышения и с повышением точности). Вычислим физические значения результата в первом варианте алгоритма :

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-2} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-8}}{2^{-9}} =$$
$$= 2^7 + 2^6 + 2^4 + 2^3 + 2^1 = 218 = 234 - 16$$

Нетрудно видеть, что и во втором варианте будет получен тот же самый результат.

- *Пример 3*. Возьмем следующие физические значения исходных операндов : X = -234; Y = -16.

Машинные представления операндов соответственно будут выглядеть (после перевода их в дополнительный код) так : 8B00 0000 *h* и C000 0000 *h*.

Полученные машинные представления результатов (по первому и второму варианту алгоритма) : C1800 0000 *h* и 8300 0000 *h*. Переводя эти значения в прямой код (получим [1.011 1110 1000 0000…0000] и [1.111 1101 0000…0000]) и производя их деление на соответствующие значения масштабных коэффициентов, получим результаты, аналогичные полученным в примере 1 данного пункта, только со знаком «минус».

- *Пример 4*. Возьмем такие машинные значения исходных операндов, чтобы соответствующие им физические значения были дробными. Дробную часть числа формируют единицы в разрядах его машинного представления, начиная с *девятого* (считая слева направо) из старших разрядов мантиссы – для $\overline{X}$ и с *шестого* – для $\overline{Y}$. (так как абсолютные значения показателей степени масштабных коэффициентов равны соответственно 8 и 5). Положим целые части физических представлений чисел равными максимальным за вычетом единицы (соответственно равными 233 и 15).

$$\overline{X}_{\text{цел}} = X_{\text{цел}} \cdot \mathcal{M}_X = \left(2^7 + 2^6 + 2^5 + 2^3 + 2^0\right) \cdot 2^{-8} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-8};$$

$$\overline{Y}_{\text{цел}} = Y_{\text{цел}} \cdot \mathcal{M}_Y = 2^{-5}\left(2^3 + 2^2 + 2^1 + 2^0\right) = 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5}.$$

Сравним результаты выполнения операции с обычной и повышенной точностью :

$$\overline{X} = \left[\tilde{0}111\ 0100\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\right] = 74FF\ FFFF\ h;$$

$$\overline{Y} = \left[\tilde{0}011\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\right] = 3FFF\ FFFF\ h;$$

$$\overline{X} \cdot 2^{-1} = \left[\tilde{0}111\ 1010\ 0111\ 1111\ 1111\ 1111\ 1111\ 1111\right] = 3A7F\ FFFF\ h$$

$$\overline{Y} \cdot 2^{-4} = \left[\tilde{0}000\ 0011\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\right] = 03FF\ FFFF\ h;$$

$$\overline{X} \cdot 2^{-1} + \overline{Y} \cdot 2^{-4} = \left[\tilde{0}011\ 1110\ 01\mathbf{11\ 1111\ 1111\ 1111\ 1111\ 1110}\right] = 3E7F\ FFFE\ h.$$

$$\overline{Y} \cdot 2^{-3} = \left[\tilde{0}000\ 0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\right] = 07FF\ FFFF\ h;$$

$$\overline{X} + \overline{Y} \cdot 2^{-3} = \left[\tilde{0}111\ 1100\ 1\mathbf{111\ 1111\ 1111\ 1111\ 1111\ 1110}\right] = 7CFF\ FFFE\ h.$$

В первом случае дробную часть результата сформируют биты его мантиссы, начиная с десятого (они выделены жирным), а во втором случае – начиная с девятого (они также выделены жирным). Во втором случае дробная часть результата окажется равной

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-9} + \ldots + 2^{-30}}{2^{-8}} = 2^{-1} + \ldots + 2^{-22},$$

тогда как в первом -

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-10}+...+2^{-30}}{2^{-9}} = 2^{-1}+...+2^{-21}.$$

Как видим, в данном примере удалось реально (пусть и не существенно) повысить точность результата (в представлении результата с повышенной точностью учтено слагаемое 2^{-22}).

3.2. Операция умножения

3.2.1. Алгоритм выполнения операции : $\overline{Z} = \overline{X} * \overline{Y}$**.** Так как каждый из операндов содержит по знаковому биту, а в микропроцессорах фирмы INTEL осуществляется умножение младшими разрядами вперед, то результат содержит *по крайней мере* один незадействованный бит старшего разряда.

3.2.2. Фрагмент программы на языке ассемблера, реализующий алгоритм умножения машинных представлений чисел

```
xor   ax, ax   ;2 байта, 1 такт[1]. Обнуление регистра-приемника
mov   al, X    ;2 байта, 1 такт[2]. Запись первого операнда в младшую часть приемника
mov   bl, Y    ; 2 байта, 1 такт. Запись второго операнда в отдельный регистр
imul  bl       ; 2 байта, 13..17 тактов. Знаковое умножение первого операнда на  второй
mov   Z, ah    ; 2 байта, 1 такт. Запись результата в ячейку основной памяти
…
```

Для повышения эффективности использования разрядной сетки и точности выполнения операции можно дополнительно использовать один или два незадействованных разряда результата (в нашем случае – два, так как $\log_2\left(\frac{\mathcal{M}_Z^*}{\mathcal{M}_Z}\right) = \log_2\left(\frac{2^{-12}}{2^{-14}}\right) = 2$), где $\mathcal{M}_Z^*$ - масштабный коэффициент результата, вычисленный исходя из максимума результата. При этом вместо последней строки приведенного выше фрагмента необходимо записать следующие две :

```
…
sal   ax, 2  ; 2 байта, 1 такт.  Сдвиг результата на два незадействованных разряда (влево)
mov   Z, ah  ; 2 байта, 3 такта. Запись результата в ячейку основной памяти
…
```

3.2.3. Экспериментальный анализ операции умножения

- *Пример 1*. Максимальные значения операндов *X* и *Y*. Имеем (после черты приведены результаты алгоритма с повышенной точностью) :

[1] Все сведения по затратам оборудования на хранение команд в памяти, а также по количеству машинных тактов на их исполнение приводятся для процессора Intel 80486.

[2] В случае задания второго операнда в виде непосредственного значения количество тактов остается тем же.

$\overline{X}$ = $[\tilde{0}111\ 0101] = 75\ h$;
$\overline{Y}$ = $[\tilde{0}100\ 0000] = 40\ h$;
$\overline{X} * \overline{Y}$ = $[\tilde{0}001\ 1101\ 0100\ 0000] = 1D\ 40h$.
$\overline{Z}$ = $[\tilde{0}001\ 1101] = 1Dh$.

– –

$\overline{X} * \overline{Y}$ = $[\tilde{0}111\ 0101\ 0000\ 0000] = 75\ 00h$.
$\overline{Z}$ = $[\tilde{0}111\ 0101] = 75h$.

Можно сделать следующие выводы : во-первых, не произошло переполнения разрядной сетки в случае максимального результата; во-вторых, удалось повысить точность результата благодаря сдвигу на 2 разряда влево. Абсолютно точный результат, получаемый в данной операции, равен 234*16 = 3744. Оценим полученные результаты и относительные погрешности их вычисления :

- Без повышения точности :

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-3} + 2^{-4} + 2^{-5} + 2^{-7}}{2^{-14}} = 2^{11} + 2^{10} + 2^{9} + 2^{7} = 3712;$$

$$\delta Z = \frac{|Z_{\text{реал.}} - Z_{\text{этал.}}|}{Z_{\text{этал.}}} \cdot 100\% = \frac{3744 - 3712}{3744} \cdot 100\% \approx 0{,}86\%.$$

- С повышением точности :

$$Z = \frac{\overline{Z}}{\mathcal{M}_Z} = \frac{2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}}{2^{-12}} = 2^{11} + 2^{10} + 2^{9} + 2^{7} + 2^{5} = 3744;$$

$$\delta Z = \frac{|Z_{\text{реал.}} - Z_{\text{этал.}}|}{Z_{\text{этал.}}} \cdot 100\% = \frac{3744 - 3744}{3744} \cdot 100\% = 0\%.$$

Таким образом, в случае реализации алгоритма с повышенной точностью результат оказался не только относительно, но и даже абсолютно точным (в рассматриваемом примере)!

К сожалению, далеко не все даже целочисленные операнды из рассматриваемых диапазонов X и Y могут быть представлены абсолютно точно в имеющейся восьмиразрядной сетке, что, естественно, негативным образом сказывается на точности результата и даже может в отдельных случаях привести к нулевому результату. Например, значение 15 переменной X не может быть представлено абсолютно точно в данной разрядной сетке:
$\overline{X} = X \cdot \mathcal{M}_X = (2^0 + 2^1 + 2^2 + 2^3) \cdot 2^{-8} = 2^{-8} + 2^{-7} + 2^{-6} + 2^{-5}$. Слагаемое 2^{-8} не может быть представлено в восьмиразрядной сетке.

Пример 2. Максимальные абсолютные значения операндов X и Y, но второй операнд - отрицательный (см. пример 2 операции сложения). С учетом знака исходные операнды, подлежащие занесению в ЭВМ : $75\ h$ и $C0\ h$ (дополнительный код).

Осуществив прогон программы с указанными машинными представлениями операндов, получим машинные результаты: $E2h$ = $\left[\tilde{1}\,110\,0010\right]$ и $8Bh$=$\left[\tilde{1}\,000\,1011\right]$ (соответственно по алгоритму без повышения и с повышением точности). Переведя эти значения в прямой код, получим соответственно $\left[\tilde{1}\,001\,1101\right]+\left[0...01\right]=\left[\tilde{1}\,001\,1110\right]$ и $\left[\tilde{1}\,111\,0100\right]+\left[0...01\right]=\left[\tilde{1}\,111\,0101\right]$. Физическое значение результата в варианте алгоритма с повышением точности совпадает с вычисленным в предыдущем примере, но отличается знаком (- 3744); в случае без повышения точности нами получен результат : - 3840, что составляет 2,56 % относительной погрешности.

Пример 3. Максимальные абсолютные значения операндов *X* и *Y*, но оба операнда - отрицательны (см. пример 3 операции сложения). С учетом знака исходные операнды, подлежащие занесению в ЭВМ : $8Bh$ и $C0h$ (дополнительный код).

Осуществив прогон программы с указанными машинными представлениями операндов, получим машинные результаты: ($1Dh$ и $75h$ соответственно по алгоритму без повышения и с повышением точности). Эти значения положительны и полностью совпадают с полученными в примере 1 настоящего пункта.

Пример 4. Максимальные абсолютные значения операндов *X* и *Y*, но первый операнд - отрицательный (см. пример 2 операции сложения). С учетом знака исходные операнды, подлежащие занесению в ЭВМ : $8Bh$ (дополнительный код) и $40h$.

Осуществив прогон программы с указанными машинными представлениями операндов, получим машинные результаты: $E2h$ и $8Bh$ (соответственно по алгоритму без повышения и с повышением точности), что совпадает со значениями, полученными в примере 2.

3.2.4. Построение графика зоны нулевого результата для операции умножения

Вначале получим теоретически аналитическое выражение для графика нулевой зоны результата. Сразу оговоримся, что получаемая формула верна для всех машинных значений операндов из заданных диапазонов, но только для тех физических значений из диапазонов, которые могут быть абсолютно точно представлены в восьмиразрядной сетке. Будем рассматривать случай с повышенной точностью результата. Все операнды для простоты будем считать положительными (это нисколько не умаляет общности результата).

Нулевой результат (даже при ненулевых машинных исходных значениях) может возникнуть вследствие того, что при определенных сочетаниях исходных операндов старшая часть регистра-приемника, которая и рассматривается как результат, состоит из одних нулей (за исключением, быть может, знакового бита). Для получения нулевого результата должно выполняться соотношение :

$$\overline{Z} < 2^{-7};$$
$$\overline{X} \cdot \overline{Y} \cdot 2^{-1} \cdot 2^{2} < 2^{-7};$$
$$\overline{Y} < \frac{2^{-8}}{\overline{X}}.$$

Для всех физических значений операндов, представляющихся полностью в разрядной сетке, можно далее получить неравенство зоны нулевого результата:

$$X \cdot \mathcal{M}_X \cdot Y \cdot \mathcal{M}_Y < 2^{-8};$$
$$X \cdot Y < 2^{5};$$
$$Y < \frac{32}{X}.$$

График, построенный по приведенным выше выражениям, имеет вид :

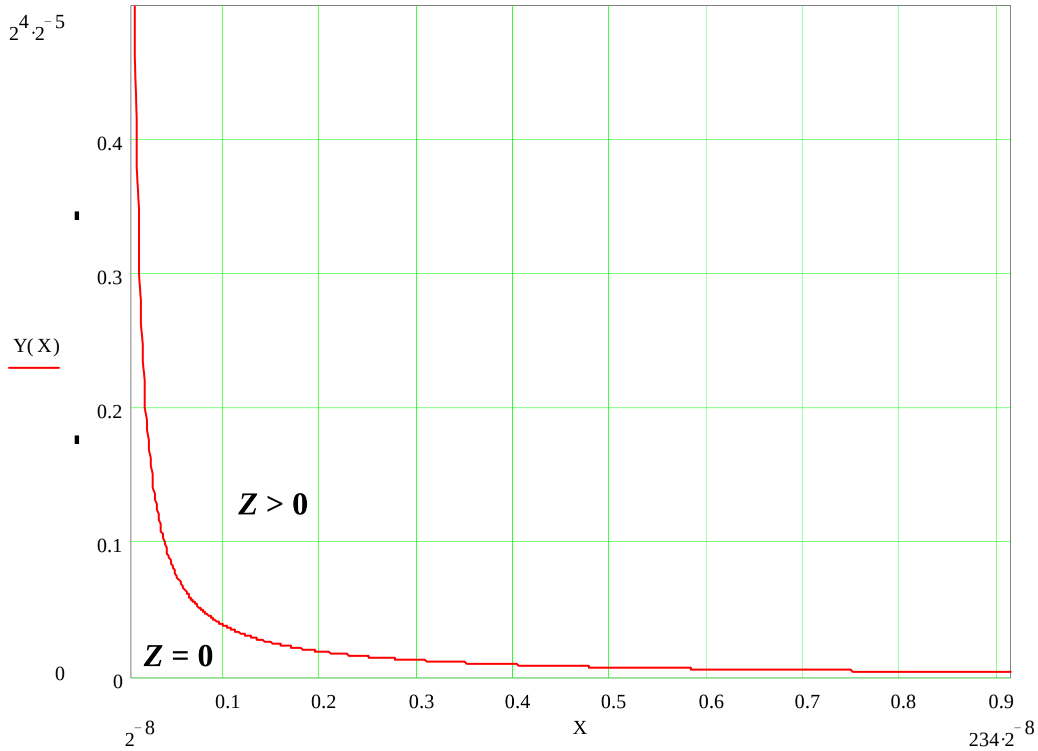


Рис. 3.1. График зоны нулевого результата операции умножения

По осям X и Y отложены машинные представления аргументов.

Приведем теперь несколько примеров, подтверждающих справедливость построенного графика :

- Пример 1 пункта 3.2.3 : произведение исходных операндов равно $3744 > 32$ (физические представления); машинный результат операции, как и следует ожидать, больше нуля.

- X=16, Y=2; В данном случае имеем : $XY = 32$; $\overline{X} = X \cdot \mathcal{M}_X = 2^4 \cdot 2^{-8} = 2^{-4}(08h)$;$\overline{Y} = Y \cdot \mathcal{M}_Y = 2^1 \cdot 2^{-5} = 2^{-4}(08h)$;

  $\overline{X} \cdot \ \overline{Y} = 2^{-8}$(как видим, такое соотношение операндов соответствует точке на разделяющей кривой графика). Получили результат, отличный от нуля : $01h$.

- Y=16, X=2; В данном случае имеем : $XY = 32$; $\overline{X} = X \cdot \mathcal{M}_X = 2^1 \cdot 2^{-8} = 2^{-7}(01h)$;$\overline{Y} = Y \cdot \mathcal{M}_Y = 2^4 \cdot 2^{-5} = 2^{-1}(40h)$;

  $\overline{X} \cdot \ \overline{Y} = 2^{-8}$(как видим, такое соотношение операндов соответствует точке на разделяющей кривой графика). Получили результат, отличный от нуля : $01h$.

- X=2, Y=15,75; В данном случае имеем : $XY = 31{,}5 < 32$; $\overline{X} = X \cdot \mathcal{M}_X = 2^1 \cdot 2^{-8} = 2^{-7}(01h)$;

  $\overline{Y} = 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7}(3\text{F}h)$;

  Получили результат, равный нулю, что и следовало ожидать.

- X=4, Y=8; В данном случае имеем : $XY = 32$; $\overline{X} = 2^{-6}(02h)$;$\overline{Y} = 2^{-2}(20h)$;
  $\overline{X} \cdot \ \overline{Y} = 2^{-8}$ (как видим, такое соотношение операндов соответствует точке на разделяющей кривой графика). Получили результат, отличный от нуля : $01h$.

- X=8, Y=4; В данном случае имеем : $XY = 32$; $\overline{X} = 2^{-5}(08h)$;$\overline{Y} = 2^{-3}(10h)$;

  $\overline{X} \cdot \ \overline{Y} = 2^{-8}$(как видим, такое соотношение операндов соответствует точке на разделяющей кривой графика). Получили результат, отличный от нуля : $02h$.

- X=32, Y=1; В данном случае имеем : $XY = 32$; $\overline{X} = 2^{-3}(10h)$;$\overline{Y} = 2^{-5}(04h)$; $\overline{X} \cdot \ \overline{Y} = 2^{-8}$ (как видим, такое соотношение операндов соответствует точке на разделяющей кривой графика). Получили результат, отличный от нуля : $01h$.

- X=8, Y=3,75; В данном случае имеем : $XY = 30 < 32$; $\overline{X} = 2^{-5}(04h)$;

  $\overline{Y} = 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7}(0\text{F}h)$;

  Получили результат, равный нулю, что и следовало ожидать.

- X=4, Y=7,75; В данном случае имеем : $XY = 31 < 32$; $\overline{X} = 2^{-6}(02h)$;

  $\overline{Y} = 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7}(1\text{F}h)$;

  Получили результат, равный нулю, что и следовало ожидать.

- X=32, Y=0,75; В данном случае имеем : $XY = 24 < 32$; $\overline{X} = 2^{-3}(10h)$;

  $\overline{Y} = 2^{-6} + 2^{-7}(03h)$;

  Получили результат, равный нулю, что и следовало ожидать.

- X=16, Y=1,75; В данном случае имеем : $XY = 28 < 32$; $\overline{X} = 2^{-4}(08h)$;
  $\overline{Y} = 2^{-5} + 2^{-6} + 2^{-7}(07h)$;

  Получили результат, равный нулю, что и следовало ожидать.

3.3. Операция умножения на постоянную

3.3.1. Алгоритмы выполнения операции : так как мы имеем постоянную, не являющуюся степенью двойки, возможны 3 алгоритма :

- Принимаем результат равным переменному операнду ($\overline{Z} = \overline{X}$), а масштабный коэффициент результата вычисляем как отношение масштабного коэффициента этого операнда к самой постоянной (см. п. 2.4.1, формула 2).

- $\overline{Z} = \overline{X} * \overline{Y}$. В данном случае умножение осуществляется обычной командой IMUL (см. п.3.2).

- *Способ сдвига и сложения*. Имеем : $\overline{Z} = \overline{A} \cdot \overline{X} = (2^{-1} + 2^{-5}) \cdot \overline{X} = \overline{X} \cdot 2^{-1} + \overline{X} \cdot 2^{-5} = 2^{-1}(\overline{X} + \overline{X} \cdot 2^{-4})$. Теоретически можно показать, а экспериментально – подтвердить, что не возникнет переполнения в случае, если мы не будем осуществлять сдвиг всего результата вправо на один разряд (с учетом максимального значения переменной), что позволит повысить точность результата. В этом случае масштабный коэффициент результата будет вычисляться по формуле (3), а собственно алгоритм будет выглядеть следующим образом :

$$\overline{Z}_1 = \overline{X} \cdot 2^{-4};$$
$$\overline{Z} = \overline{X} + \overline{Z}_1.$$

3.3.2. Фрагмент программы на языке ассемблера, реализующий алгоритм умножения на постоянную методом сдвига и сложения

```
…
mov   al, x   ;2 байта, 1 такт. Запись в регистр переменной
mov   bl, al  ;2 байта, 1 такт. Копирование переменной
sar   al, 4   ;2 байта, 3 такта. Получение промежуточного слагаемого
add   al, bl  ;2 байта, 1 такт. Сложение переменной и промежуточного слагаемого
mov   z, al   ; 2 байта, 1 такт. Запись байта результата в ячейку основной памяти
…
```

3.3.3. Сравнительный количественный анализ алгоритмов умножения на константу обычным способом и методом сдвига и сложения

Проставив возле каждого оператора ассемблерного фрагмента умножения и умножения на константу число занимаемых байт и машинных тактов исполнения, получим следующие суммарные результаты (они справедливы для процессора 80486):

- умножение обычным способом : 10 байт, 17..21 машинных тактов;
- умножение с повышением точности : 14 байт, 21..25 машинных тактов;
- умножение методом сдвига и сложения : 10 байт, 7 машинных тактов.

Таким образом, самым экономичным в смысле времени исполнения является метод сложения и сдвига. Дополнительные затраты оборудования в методе с повышением точности обусловлены наличием дополнительной команды арифметического сдвига по сравнению с обычным способом.

3.3.4. Экспериментальный анализ выполнения операции умножения на константу

Пример. Возьмем максимальное значение переменной (234). Машинное представление этого числа равно $75h$. Машинное представление константы равно $44h$. В результате прогона фрагмента программы получим результат : $7Ch$. Такой же результат можно получить (с учетом того, что второй операнд – константа со своим масштабным коэффициентом, см. п.2.4). Напомним, что масштабный коэффициент результата в этих случаях равен 2^{-13} (см. п.2.4). Реализация операции по обычному способу дает машинный результат : $1Fh$. Масштабный коэффициент результата в этом случаях равен 2^{-15} . Таким образом, можно заключить, что в случае реализации по алгоритму умножения с повышением точности и по методу сдвига и сложения получаем более точные результаты за счет дополнительного использования двух старших разрядов сетки.

3.4. Операция деления

3.4.1. Алгоритм выполнения операции :

$$\overline{X}_1 = \overline{X} \cdot 2^{-N};$$

$$\overline{Z} = \frac{\overline{X}_1}{\overline{Y}}$$

3.4.2. Фрагмент программы на языке ассемблера :

```
…
mov   ax, X   ; 2 байта, 1 такт. Поместить операнд-делимое в приемник
cwd           ; 2 байта, 3 такта. Расширение приемника знаковым битом
mov   bx, Y   ; 2 байта, 1 такт. Запись в регистр делителя
idiv  bx      ; 2 байта, 27 тактов. Собственно знаковое деление
mov   Z, ax   ; 2 байта, 1 такт. Запись результата в оперативную память
…
```

3.4.3. Экспериментальный анализ операции

Пример 1. Выберем операнды таким образом, чтобы результат получился максимальным. При этом делимое должно быть максимальным, а делитель –

минимальным из тех, которые только могут быть представлены в 16-тиразрядной сетке:

$$X = 234,\ \overline{X} = 7500h;\ \overline{Y} = +2^{-15} = 0001h;\ Y = 2^{-10}.$$

В результате получаем следующее: $\overline{Z} =$ 0111 0101 0000…0000 = 7500 0000 h. Переводя это число в физическое представление, получаем (с учетом масштабного коэффициента результата 2^{-18}):

$$2^{18}\left(2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}\right) = 2^{17} + 2^{16} + 2^{15} + 2^{13} + 2^{11} = 239\,616.$$

Это – максимальное число, которое может быть получено в данной операции. Расчеты (234 * 1024 = 239616) показывают, что число получено абсолютно точным.

3.4.4. Построение графика зоны нулевого результата для операции деления

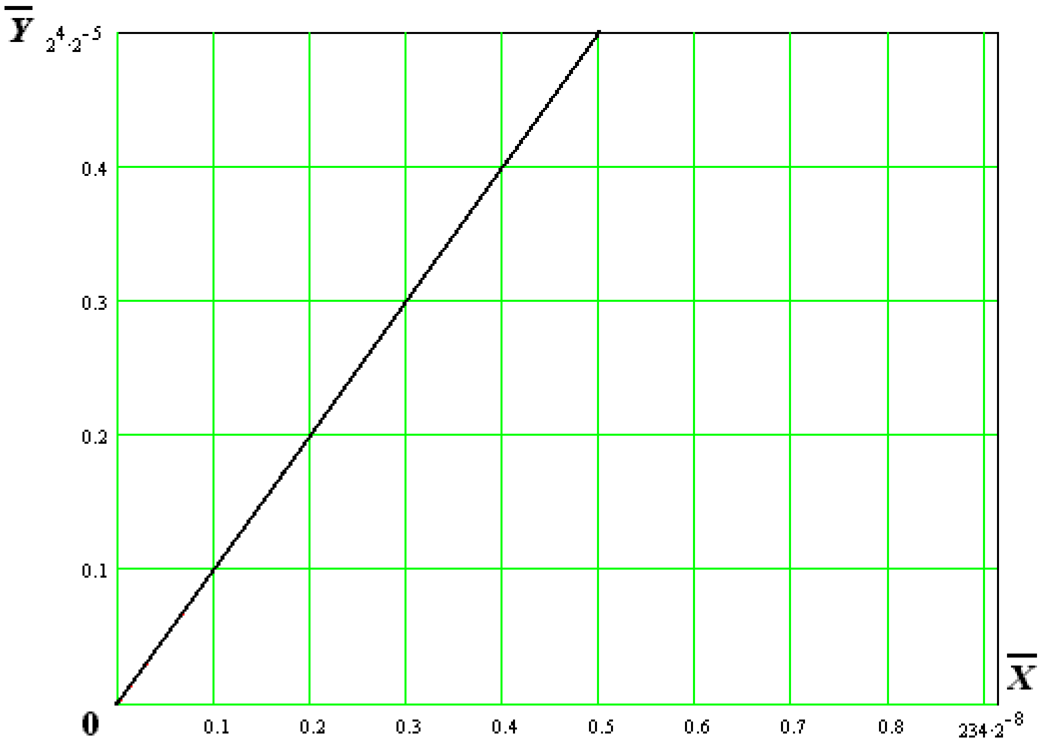
Нулевой результат (даже при ненулевых машинных исходных значениях) может возникнуть вследствие того, что приемник, который и рассматривается как результат, состоит из одних нулей (за исключением, быть может, знакового бита). Для получения нулевого результата должно выполняться соотношение :

$$\begin{aligned} &\overline{Z} < 2^{-15}; \\ &\frac{\overline{X} \cdot 2^{-15}}{\overline{Y}} < 2^{-15}; \\ &\overline{X} < \overline{Y}. \end{aligned} \qquad (4)$$

Для всех физических значений операндов, представляющихся полностью в разрядной сетке, можно далее получить неравенство зоны нулевого результата:

$$\begin{aligned} &X \cdot \mathcal{M}_X < Y \cdot \mathcal{M}_Y; \\ &X \cdot 2^{-8} < Y \cdot 2^{-5}; \\ &Y > X \cdot 2^{-3}. \end{aligned} \qquad (4^*)$$

График, построенный по приведенным выше выражениям, имеет вид прямой, проходящей через начало координат (на его осях проставлены машинные значения):



Приведем примеры :

- В примере 1 $X = 234,\ Y = 2^{-10}$, т.е. условие нулевой зоны (4^*) не выполняется, следовательно, результат должен был получиться ненулевой, что и имеет место.
- Рассмотрим случай Y = 16, X = 128. Это соответствует машинным значениям операндов : $\overline{X} = 2^{-1}(4000h), \overline{Y} = 2^{-1}(4000h)$. Получили в AX результат, отличный от нуля : $0001h$, что и следовало ожидать (этот результат располагается на самой кривой, разделяющей области нулевого и ненулевого результата).
- Рассмотрим случай Y = 8, X = 64. Это соответствует машинным значениям операндов : $\overline{X} = 2^{-2}(2000h), \overline{Y} = 2^{-2}(2000h)$. Получили в AX результат, отличный от нуля : $0001h$, что и следовало ожидать (этот результат располагается на самой кривой, разделяющей области нулевого и ненулевого результата).
- Рассмотрим случай Y = 4, X = 32. Это соответствует машинным значениям операндов : $\overline{X} = 2^{-3}(1000h), \overline{Y} = 2^{-3}(1000h)$. Получили в AX результат, отличный от нуля : $0001h$, что и следовало ожидать.
- Рассмотрим случай Y = $16{+}2^{-10}$, X = 128. Это соответствует машинным значениям операндов: $\overline{X} = 2^{-1}(4000h), \overline{Y} = 2^{-1} + 2^{-15}(4001h)$. Получили в AX результат, равный нулю : $0001h$, что и следовало ожидать, так как выполняется условие (4^*)).

- Рассмотрим случай $Y = 8+2^{-10}$, $X = 64$. Это соответствует машинным значениям операндов : $\overline{X} = 2^{-2}(2000h), \overline{Y} = 2^{-2} + 2^{-15}(2001h)$. Получили в AX результат, отличный от нуля : $0001h$, что и следовало ожидать, так как выполняется условие (4*)).
- Рассмотрим случай $Y = 4+2^{-10}$, $X = 32$. Это соответствует машинным значениям операндов : $\overline{X} = 2^{-3}(1000h), \overline{Y} = 2^{-3} + 2^{-15}(1001h)$. Получили в AX результат, отличный от нуля : $0001h$, что и следовало ожидать.

3.5. Операция деления постоянной на переменную

3.5.1. Алгоритм выполнения операции : данная операция не имеет никаких особенностей в реализации и осуществляется аналогично общему случаю деления переменной на переменную, за тем исключением, что при расчете масштабного коэффициента для постоянной ее величина рассматривается как максимальная величина возможной переменной (см.п.2.6).

3.5.2. Экспериментальный анализ операции

Для примера возьмем минимальное значение знаменателя, чтобы получить наибольший результат. Имеем :

$$A = 34;\ \overline{A} = 2^{-1} + 2^{-5} = 4400h;\ \overline{X} = 2^{-15};\ X = 2^{-7}.$$

В результате получаем следующее: $\overline{Z}$ = 0100 0100 0000 0000 = $4400h$. Как видим, результат вошел в разрядную сетку без переполнения. Переводя это число в физическое представление, получаем (с учетом масштабного коэффициента результата):

$$Z = \frac{2^{-1} + 2^{-5}}{2^{-13}} = 2^{12} + 2^{8} = 4352.$$

Расчеты ($34 * 128 = 4352$) показывают, что число получено абсолютно точным.

3.6. Операция деления переменной на постоянную

3.6.1. Алгоритм выполнения операции : операцию можно реализовывать обычным путем и методом с повышенной точностью результата. Во втором случае выполняются следующие действия :

- Обнуляется приемник (аккумуляторный регистр, AX);
- В его расширение (DX) загружается делимое;
- Осуществляется сдвиг в приемнике вправо на число разрядов, равное

$$r = N - \log_2\left(\frac{\mathcal{M}_Z^*}{\mathcal{M}_Z}\right) = 16 - \log_2\left(\frac{2^{-5}}{2^{-19}}\right) = 16 - 14 = 2. \text{ (см. п.2.7)}$$

3.6.2. Фрагмент ассемблерной программы реализации операции деления переменной на постоянную :

```
…
xor    ax, ax ;2 байта, 1 такт. Обнулить регистр-аккумулятор
mov    dx, X  ; 2 байта, 1 такт. Загрузка делимого в расширение аккумулятора
mov    bx, B  ; 2 байта, 1 такт. Загрузка делителя
sar    dx, 1  ; 1 байт, 3 такта. Сдвиг во флажок переноса содержимого расширения
rcr    ax, 1  ; 1 байт, 3 такта. Запись содержимого флажка переноса в старший бит акк.
sar    dx, 1  ; 1 байт, 3 такта.
rcr    ax, 1 ;  1 байт, 3 такта.
idiv   bx     ; 2 байта, 27 тактов. Собственно знаковое деление
mov    Z, ax ; 2 байта, 1 такт.
…
```

3.6.3. Сравнительный количественный анализ алгоритмов деления на константу обычным способом и способом с повышением точности

Проставив возле каждого оператора ассемблерного фрагмента деления и деления на константу число занимаемых байт и машинных тактов исполнения, получим следующие суммарные результаты (они справедливы для процессора 80486):

- деление обычным способом : 10 байт, 33 машинных такта;
- деление с повышением точности : 14 байт, 43 машинных такта.

Таким образом, плата за повышение точности в данном случае – увеличение занимаемой командами памяти (на 4 байта) и времени исполнения операции (на 10 машинных тактов).

3.6.4. Экспериментальный анализ алгоритмов деления на константу

Возьмем для примера максимальное значение делимого для того, чтобы получить максимальный результат. Имеем :

$$X = 234; \overline{X} = +\left(2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}\right) = 7500h;$$
$$B = 14{,}2; \overline{B} = 2^{-4}\left(2^{3} + 2^{2} + 2^{1} + 2^{-3} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-11}\right) =$$
$$= 2^{-1} + 2^{-2} + 2^{-3} + 2^{-7} + 2^{-8} + 2^{-11} + 2^{-12} + 2^{-15} = 7199h.$$

После прогона фрагмента программы по алгоритму без повышения точности получим машинный результат, равный $\overline{Z} = 0001h = 2^{-15}$. Поэтому $Z = \dfrac{\overline{Z}}{\mathcal{M}_Z} = \dfrac{2^{-15}}{2^{-19}} = 2^4 = 16$, тогда как истинный результат равен $16{,}4788732394366197183098591549296\ldots$ Как видим, в данном случае пустует почти вся разрядная сетка (за исключением самого младшего бита).

Алгоритм с повышением точности дает машинный результат, равный $\overline{Z} = 41\mathrm{EA}h = 2^{-1} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-10} + 2^{-12} + 2^{-14}$. Как видим, здесь разрядная сетка используется в гораздо большей мере, причем первый старший бит мантиссы – ненулевой. Вычисленное с учетом масштабного коэффициента (см. п. 2.7) физическое значение результата равно уже 16,478515625… . Как видим, точность при этом заметно возросла по сравнению с полученным выше результатом.

3.7. Операция умножения в предельных масштабах. Сравнительный анализ (в сравнении с операцией умножения в двоичных масштабах без повышения точности)

Для примера возьмем случай, когда результат получается максимальным (при этом исходные множитель и множимое также максимальны).

Случай двоичных масштабов	*Случай предельных масштабов*
$\mathcal{M}_X = 2^{-8}$	$\mathcal{M}_X = \dfrac{\overline{X}_{\max}}{\lvert X_{\max}\rvert} = \dfrac{2^{-1}+2^{-2}+2^{-3}+2^{-4}+2^{-5}+2^{-6}+2^{-7}}{2^7+2^6+2^5+2^3+2^1}$
$\mathcal{M}_Y = 2^{-5}$	$\mathcal{M}_Y = \dfrac{2^{-1}+2^{-2}+2^{-3}+2^{-4}+2^{-5}+2^{-6}+2^{-7}}{2^4}$
$\overline{X} = 75h$	$\overline{X} = 7\mathrm{F}h$
$\overline{Y} = 40h$	$\overline{Y} = 7\mathrm{F}h$
$\overline{Z} = \left[\tilde{0}001\,1101\right] = 1\mathrm{D}h.$	$\overline{Z} = \left[\tilde{0}011\,1111\right] = 3\mathrm{F}h.$
$\mathcal{M}_Z = 2^{-1}\,\mathcal{M}_X\mathcal{M}_Y = 2^{-14}$	$\mathcal{M}_Z = \mathcal{M}_X\mathcal{M}_Y 2^{-1} \approx 1{,}315\cdot 10^{-4}$
Z = 3712 (3744; 0,86%)*	Z = 3744 (3744; 0%)

Таким образом, видим, что машинный результат представлен в случае предельных масштабов с максимальной информативностью, а следовательно, и физический результат получен с максимальной точностью. Однако масштабные коэффициенты вычислять в этом случае гораздо сложнее.

* В скобках указано абсолютно точное (эталонное) значение результата и его относительная погрешность