Sprawozdanie

Koncepcja reprezentacji liczb wymiernych w procesorze RNS

Marcin Wojciechowski 235088
Prowadzący: Dr inż. Piotr Patronik
Termin zajęć: piątek, TN, 17:05

I. Wstęp

Głównym celem projektu było zapoznanie się z koncepcją reprezentacji liczb wymiernych w procesorze bazującym na resztowym systemie liczbowym. Koncepcja ta została zaproponowana przez Eric'a B. Olsena, który opatentował swoje rozwiązanie.

Kandydatem na zastępcę dla systemu binarnego w procesorze jest system resztowy (skr. RNS). Nie wymaga on obsługi przeniesień, przez co podstawowe działania jak dodawanie, odejmowanie oraz mnożenie mogą wykonywać się szybciej.

II. Przebieg pracy

- ### Zajęcia pierwsze

Podczas pierwszych zajęć wybrany został temat. Czynnikami decydującymi były:

- Posiadanie podstaw teoretycznych dla systemu resztowego realizowane w trakcie kursu Archutektura komputerów 1
- Chęć poznania alternatywnej koncepcji architektury jednostki obliczeniowej (nie bazującej na systemie binarnym)

- ### Zajęcia drugie

Podczas prac w drugim tygodniu, zacząłem poznawać koncepcję reprezentacji liczb wymiernych w RNS.

Koncepcja reprezentacji liczb wymiernych

Reprezentacja zaproponowana przez E. B. Olsena wygląda następująco:

Równanie 1 $$I_1, I_2, \ldots, I_M, F_1, F_2, \ldots, F_N$$

gdzie liczby I_1 do I_M oznaczają moduły RNS zarezerwowane dla części całkowitej liczby, a liczby F_1 do F_N oznaczają moduły RNS zarezerwowane dla części ułamkowej liczby.

Liczby zapisane w powyższej reprezentacji (M+N) traktowane są jako jedna liczba RNS. Oznacza to, że wszystkie moduły muszą być względnie pierwsze.

Aby otrzymać wartość dziesiętną liczby z tej reprezentacji, wykonujemy konwersję otrzymując wartość liczby zapisanej w systemie RNS o wektorze bazowym a następnie dzielimy tą wartość przez iloczyn modułów ułamkowych (ang. fractional range):

Równanie 2 $$R_F = (F_1 * F_2 * \ldots * F_N)$$

Przykładowo, liczba $\{2,7,16,5,5,3,2,1,1\}$ dla modułów równych $\{23,19,17,13 \cdot 11,7,5,3,2\}$, po konwersjii 577, ma wartość $\frac{577}{11*7*5*3*2} = \frac{577}{2310} = 0.2498$.

Dziesiętną liczbę 1 zapiszemy konwertując R_F na RNS. Dla powyższych przykładowych modułów:

$$1.0_{10} = R_F = \{10,11,15,9,0,0,0,0,0\}$$

Ta reprezentacja ma pewne ograniczenia. Mianowicie, liczby które możemy zapisać w tym formacie są wielokrotnością następującej liczby:

$$ump = 1/(F_1 * F_2 * \ldots * F_N)$$

co oznacza, że przykładowo dla części ułamkowej o modułach $\{2, 3, 5, 7, 11\}$ najmniejszą liczbą jaką możemy zapisać jest $\frac{1}{2310} = 0{,}0004329$, a każda kolejna wartość będzie wielokrotnością tej liczby.

Zakres liczb całkowitych (ang. whole range), czyli największa liczba całkowita, którą możemy zapisać w tym systemie jest równa:

Równanie 3 $$R_W = (I_1 * I_2 * \ldots * I_M)$$

- Zajęcia trzecie i czwarte

Praca w kolejnych tygodniach opierały się na zrozumieniu metody mnożenia w stałoprzecinkowym systemie resztowym.
Aby zrozumieć koncepcję mnożenia, konieczne było poznanie systemu zmiennobazowego (ang. mixed-radix, skr. MR), konwersji z systemu RNS do dziesiętnego przy wykorzystaniu systemu MR (ang. Mixed-Radix Conversion, skr. MRC), konwersji z RNS na system MR oraz z MR na RNS.

Koncepcja mnożenia

Powodem powyższych wymagań jest fakt, że liczby we wspomnianej powyżej koncepcji zapisane są w postaci $X = \frac{X_i}{R_f}$, gdzie X_i jest liczbą interpretowaną przez RNS o zwyczajnych modułach. Jeśli zdefiniujemy również liczby $Y = \frac{Y_i}{R_f}$ oraz $Z = \frac{Z_i}{R_f}$ i będziemy chcieli policzyć $X \cdot Y = Z$, otrzymamy następujące równanie:

$$Z_i = \frac{X_i Y_i}{R_f}$$

Oznacza to, że wynik mnożenia liczb musi zostać podzielony przez zakres liczb ułamkowych. Wspomniane podzielenie wyniku najprościej wykonać w systemie MR.

System zmiennobazowy

W systemie zmiennobazowym, liczby reprezentowane są jako wektor, w którym każda kolejna liczba jest wielokrotnością następnej, mniejszej jednostki.

Przykładowo, czas 3 dni, 7 godzin i 15 minut w systemie zmiennobazowym możemy reprezentować jako: $B = \{7, 24, 60\}$, $X = \{3,7,15\}$. Obliczenie minut w tym zapisie wyglądałoby następująco:

$$3 * 24 * 60 + 7 * 60 + 15 = 4755$$

Konwersja z RNS na system dziesiętny

Konwersja z RNS na system dziesiętny bazująca na systemie MR oparta jest na następujących wzorach:

Równanie 4 $X_i = d_1 + m_1 d_2 + \cdots + M_{i-1} d_i$

Równanie 5 $X_1 = d_1 = x_1$

Równanie 6 $d_i = \left| \frac{x_i - X_{i-1}}{M_{(i-1)}} \right|_{m_i}$

d_i oznacza i-tą cyfrę systemu MR,
x_i oznacza i-tą cyfrę RNS,
X_i oznacza wynik i-tej iteracji konwersji (X_N, gdzie N oznacza długość wektora bazowego RNS, oznacza wynik końcowy)

Konwersja z RNS na system MR

W procesorze E. B. Olsena, konwersja z RNS na system MR została zaimplementowana przy użyciu stosu. Na stos wpychane są kolejno moduły MR oraz wartości odpowiadające danym modułom. Zrealizowane jest to przez dany algorytm:
d_i oznacza liczbę RNS o i-tym indeksie
M_i oznacza moduł RNS o i-tym indeksie

1. Zacznij iterację od najmniejszego modułu (i = 0),
2. Jeśli moduł jest oznaczony jako pominięty, przejdź do punktu 9,
3. Wepchnij d_i na stos,
4. Jeśli d_i jest różna od 0 odejmij ją od kazdej nie pominiętej liczby RNS,
5. Jeśli wszystkie liczby RNS są równe 0, przejdź do punktu 11,
6. Zaznacz M_i jako pominięty,
7. Pomnóż liczbę RNS przez odwrotność modularną M_i,
8. Wepchnij M_i na stos,
9. Zwiększ i,
10. Przejdź do punktu 2,
11. Koniec.

Konwersja z systemu MR na RNS

Konwersja odwrotna do przedstawionej powyżej jest bardzo podobna. Wykorzystujemy przy tym stos zbudowany w tej samej postaci co powyżej (liczba, moduł, liczba, …, moduł, liczba) oraz liczbę RNS zapełnioną zerami, która na końcu będzie odpowiadać przekonwertowanej liczbie.
Konwersję wykonujemy poprzez:

1. Ściągnięcie liczby d ze szczytu stosu,
2. Dodanie d do liczby RNS,
3. Jeśli stos jest pusty – przejście do punktu 7,
4. Ściągnięcie liczby M ze szczytu stosu,
5. Pomnożenie liczby RNS przez M,
6. Przejście do kroku 1,
7. Zakończenie operacji.

Mnożenie

Aby umożliwić wykonywanie mnożenia i dzielenia w stałopozycyjnym systemie resztowym, do reprezentacji liczbowej zostały dodane dodatkowe moduły, które mogą przechowywać natychmiastowy wynik mnożenia, ponieważ jego wartość może wykraczać poza zakres liczb całkowitych.

Równanie 7

$$E_1, E_2, \ldots, E_X, I_1, I_2, \ldots, I_M, F_1, F_2, \ldots, F_N$$

Jak wspomniałem powyżej, wynik mnożenia musi zostać podzielony przez zakres liczb ułamkowych. Realizowane jest to przez konwersję na system MR, a następnie konwersję odwrotną, z pominięciem modułów odpowiadających za część ułamkową liczby RNS. Jeśli wynik nie daje się zapisać w RNS o konkretnych modułach, część ułamkowa liczby RNS zapisana w MR oraz przekonwertowana na liczbę dziesiętną jest porównywana z połową zakresu części ułamkowej – jeśli jest większa lub równa – wynik jest zwiększany o 1 w celu zaokrąglenia liczby w górę.

Algorytm mnożenia:

1. Wykonaj mnożenie liczb (pomnóż przez siebie liczby na odpowiadających sobie indeksach),
2. Przekonwertuj wynik do systemu MR,
3. Wykonaj konwersję odwrotną pomijając liczby i moduły ułamkowe ze stosu,
4. Oblicz wartość pozostałych liczb na stosie i porównaj je z połową zakresu części ułamkowej,
5. Jeśli wynik z kroku 4 był większy lub równy, dodaj 1 do każdej liczby RNS,
6. Koniec.

- ## Zajęcia piąte

Na kolejne zajęcia zostałem poproszony o skupienie się na dzieleniu liczb wymiernych w RNS oraz implementacji nauczonych w trakcie projektu w C++. W patencie na którym miałem się wzorować, dzielenie jest wykonywane przez algorytm Goldschmidta, w związku z czym dzielnik musi zostać przeskalowany, aby przyjmował wartości między 0 a 1.

Algorytm Goldschmidta

Dzielenie przy użyciu algorytmu Goldschmidta polega na iteracyjnym mnożeniu dzielnej i dzielnika przez odpowiedni współczynik tak, aby dzielnik zbliżał się do 1.

Równanie 8 $$Q = \frac{N}{D}\frac{F_1}{F_1}\frac{F_2}{F_2}\frac{F_3}{F_3}\ldots$$

Współczynnik dobierany jest poprzez równanie:

Równanie 9 $$F_{i+1} = 2 - D_i$$

Jak wspomniałem wcześniej, warunkiem koniecznym jest przeskalowanie dzielnika (w związku z czym musimy przeskalować również dzielną).

Równanie 10 $$0 < D \leq 1$$

Skalowanie

Aby umożliwić skalowanie w RNS, stworzono reprezentację, nazwaną sliding point RNS. Umożliwia ona przesunięcie zakresu ułamkowego (mając liczbę w postaci takiej jak w równaniu 7, przesunięcie w lewo skutkuje zmniejszeniem liczby, a przesunięcie w prawo jej zwiększeniem).

Skalowanie w wymiernym RNS jest podobne do skalowania w innych systemie – aby przesunąć „przecinek", mnożymy lub dzielimy liczbę przez podstawę systemu. W naszym przypadku, mając liczbę w postaci takiej jak w równaniu 7, przesunięcie części ułamkowej o jedną pozycję w lewo (włączenie I_M do części ułamkowej) skutkuje podzieleniem liczby przez moduł znajdujący się na tej samej pozycji co I_M. Podobnie z wygląda przesunięcie w prawo – liczba zostaje pomnożona przez moduł znajdujący się na pozycji F_1.
Przykładowo, biorąc moduły $M = \{5,3,2\}$ oraz liczbę $X = \{3,2,1\}$, gdzie moduł o wartości 2 odpowiada za część ułamkową, liczba X odpowiada wartości 13,5 (po konwersji otrzymujemy wartość 23, następnie musimy ją podzielić przez $R_F = 2$. Gdy włączymy moduł o wartości 3 do części ułamkowej, liczba X będzie interpretowana jako wartości $3\frac{5}{6}$ (po konwersji 23, liczbę dzielimy przez $R_F = 2 * 3 = 6$.

Algorytm Goldschmidta działa o wiele szybciej, jeśli przeskalujemy dzielnik zgodnie z następującym równaniem:

Równanie 10b $$0{,}5 \leq D \leq 1$$

Aby umożliwić zwiększenie małej liczby (mniejszej niż 0,5), reprezentacja sliding point RNS umożliwia zmianę potęgi modułu o wartości 2 (wiąże się to również ze zmianą zakresu ułamkowego R_F liczby). Zaczynając od wysokiej potęgi, jeśli ją zmniejszymy, liczba zostanie interpretowana liczba zostanie pomnożona przez 2^T, gdzie T odpowiada różnicy potęgi początkowej i końcowej.

Skalowanie liczb w patencie E. B. Olsena wiąże się w dużym stopniu ze zmodyfikowaną konwersją z RNS na system MR. Jedną z różnic jest zakończenie konwersji na module o wartości 2. Do wykonania dzielenia musimy przeskalować dzielnik do opowiedniej postaci, więc aby otrzymać poprawny wynik, konieczne jest przeskalowanie dzielniej o ten sam współczynnik. Konwersji nie zaczynamy od najmniejszego modułu, lecz występującego zaraz za nim. Z każdą iteracją zapisujemy liczbę dzielnika pod modułem 2, ponieważ ta wartość w momencie ostatniego przejścia pętli będzie wpływała na zmianę potęgi modułu. W momencie kiedy dzielnik się wyzeruje, zapisujemy numer iteracji pętli, oraz ustawiamy zakres ułamkowy w danym miejscu. W tym momencie możemy przejść do modyfikacji potęgi modułu 2 – bierzemy ostatnią liczbę występującą pod modułem 2 podczas konwersji różną od 0, a następnie sprawdzamy, ile bitów liczby binarnej jest wymagane do zapisania tej liczby. Obliczamy to za pomocą wzoru: $V = \lceil \log_2 N \rceil$, gdzie V jest nową potęgą modułu, a N jest wspomnianą ostatnią liczbą pod modułem 2. Jeśli pozycja zakresu ułamkowego zmniejszyła się, wymagane jest aby przywrócić jego pierwotną pozycję, ponieważ wtedy interpretowana liczba nie będzie spełniała założeń z równania 10b (będzie mniejsza niż 0,5). Aby to wykonać, mnożymy liczby przez moduły znajdujące się pomiędzy starą a nową pozycją zakresu ułamkowego.

Dzielenie

Gdy mamy już przeskalowane obie liczby, możemy przejść do wykonywania dzielenia.

Aby wykonać szybkie dzielenie w algorytmie Goldschmidta, musimy wykonać następujące operacje:

1. Przeskaluj dzielną N i dzielnik D tak, aby D zgadzał się z równaniem 10b,
2. Oblicz współczynnik $\mathrm{F_i} = 2 - \mathrm{D_{i-1}}$,
3. Pomnóż N oraz D przez $\mathrm{F_i}$,
4. Jeśli $\mathrm{D_i} = \mathrm{D_{i-1}}$ przejdź dalej, jeśli nie to wróć do punktu 2,
5. Zapisz N jako wynik,
6. Koniec.

Implementacja

Przedstawiana przeze mnie implementacja liczb wymiernych RNS w C++ jest zbudowana przy użyciu tablic. Zawarta jest tablica modułów, która może się zmieniać w zależnosci od potęgi modułu 2 oraz tablica liczb odpowiadających za wartość RNS.

Konstruktory mogą przyjmować wartości typu double – program przekształca wtedy podaną liczbę na najbliższe przybliżenie wartości akceptowalnej przez wymierny RNS, lub long long – liczba jest wtedy interpretowana jako zwykła wartość RNS. Działania arytmetyczne oraz porównujące dwie liczby RNS wykonywane są w funkcjach przeciążających operatory. Działania zostały wykonane na podstawie algorytmów omówionych w tym sprawozdaniu.

Dla sprawdzenia wyników, przeciążyłem również rzutowanie na liczbę double – wykonuje się wtedy konwersja z RNS do systemu MR, obliczana jest wartość dziesiętna z równania 4, oraz wynik dzielony jest przez zakres ułamkowy R_F.

III. Wnioski

Podczas realizacji projektu poszerzyłem moją wiedzę o nowe zagadnienia – m. in. podstawowe informacje dotyczące systemu zmiennobazowego, który nie jest często wspominany na polskich stronach internetowych, poznałem koncepcję reprezentacji liczb wymiernych w systemie resztowym, który domyślnie służy do przedstawiania jedynie liczb całkowitych. Dużo czasu poświęciłem na czytanie patentu jednostki arytmetyczno logicznej procesora opartej na systemie resztowym w poszukiwaniu informacji na temat liczb wymiernych RNS oraz działań wykonywanych na tych liczbach, jednakże nie udało mi się wszystkiego wykonać z powodu niewiedzy oraz braku doświadczenia w realizowaniu zadań na podstawie artykułów naukowych. Próby zrozumienia niektórych zagadnień często były nieskuteczne, lecz metodą prób i błędów oraz wielokrotnym czytaniem informacji zawartych w patencie myślę, że udało mi się opanować część z nich. Poza wiedzą, z pewnością umocniłem moją umiejętność przeszukiwania źródeł danych w celu znalezienia interesujących mnie informacji, co jeszcze w trakcie trwania semestru sprawiło mi trudność. Uważam, że doświadczenie zdobyte podczas realizacji projektu ułatwi mi pracę w trakcie studiów podczas kolejnych semestrów.

III. Źródła

[1]. Eric B. Olsen, „Residue number arithmetic logit unit", patent amerykański, US 2013/0311532A1, Listopad 21, 2013

[2]. Piotr Patronik, Stanisław J. Piestrak, „Hardware/Software Approach to Designing Low-Power RNS-Enhanced Arithmetic Units", IEEE Transactions on circuits and systems-i: Regular papers, vol. 64, no. 5, 2017