ARIFMETIK IFODALARNI MASHINAGA BOG'LIQMAS OPTIMALLASHTIRISH

Umarov Bekzod Azizovich
Farg'ona davlat universiteti
ubaumarov@gmail.com
Nabijonova Dilafruz Rafiqjon qizi
Farg'ona davlat universiteti 2-kurs talabasi
nabijonovadilafruz23@gmail.com

Anotatsiya: *Ushbu maqolada arifmetik ifodalarni mashinaga bog'liq bo'lmagan tarzda optimallashtirish usullari tahlil qilinadi. Asosiy e'tibor ifodalarni qayta yozish, ortiqcha hisoblashlarni kamaytirish, algebraik xususiyatlardan foydalanish va kodni umumiy holda soddalashtirishga qaratilgan. Mazkur yondashuv kompilatorlar va dasturiy ta'minotning samaradorligini oshirishda muhim ahamiyat kasb etadi. Mashinaga bog'liq bo'lmagan optimallashtirish usullari turli arxitekturalarda barqaror natijalar berishi bilan ajralib turadi.*

Kalit so'zlar: *Arifmetik ifoda, optimallashtirish, kompilator, algebraik soddalashtirish, samaradorlik.*

Annotation: *This article analyzes techniques for machine-independent optimization of arithmetic expressions. The focus is on expression rewriting, reducing redundant computations, leveraging algebraic properties, and simplifying code in a general form. Such optimization plays a significant role in improving the efficiency of compilers and software systems. Machine-independent methods are especially valuable for achieving stable performance across different hardware architectures.*

Keywords: *Arithmetic expression, optimization, compiler, algebraic simplification, efficiency.*

Аннотация: *В данной статье рассматриваются методы машинно-независимой оптимизации арифметических выражений. Основное внимание уделено переписыванию выражений, сокращению избыточных вычислений, использованию алгебраических свойств и общему упрощению кода. Такой подход важен для повышения эффективности компиляторов и программных систем. Методы, не зависящие от архитектуры машины, обеспечивают стабильные результаты на различных аппаратных платформах.*

Ключевые слова: *Арифметическое выражение, оптимизация, компилятор, алгебраическое упрощение, эффективность.*

Kirish

Hisoblashda mashinaga bog'liqlik ma'lum bir apparat platformasi yoki operatsion tizimiga xos bo'lgan va turli apparat yoki dasturiy ta'minot muhitlarida ko'chma bo'lmagan dastur yoki tizimning xususiyatlariga ishora qiladi.

Masalan, mashinaga bog'liq bo'lgan dastur muayyan turdagi protsessor yoki operatsion tizimga xos bo'lgan ko'rsatmalar yoki xususiyatlardan foydalanishi mumkin va boshqa turdagi apparat yoki dasturiy ta'minotda ishlamaydi. Mashinaga bog'liq bo'lgan dasturlar ko'pincha assembler tilida yoziladi, bu kompyuterning apparati bilan chambarchas bog'liq bo'lgan past darajadagi dasturlash tilidir.

Mashinaga bog'liqmas deganda, kompyuter yoki kompyuter uskunasining ma'lum bir turiga bog'liq bo'lmagan yoki bog'liq bo'lmagan narsa tushuniladi. U dasturiy ta'minotni, dasturlash tillarini, ma'lumotlar formatlarini yoki turli xil apparat platformalarida o'zgartirish yoki maxsus moslashtirishni talab qilmasdan foydalanish mumkin bo'lgan kompyuter tizimlarining boshqa elementlarini tavsiflash uchun ishlatilishi mumkin.

Mashinadan bog'liqmas ko'p turdagi dasturiy ta'minot va boshqa kompyuterga asoslangan tizimlar uchun muhim xususiyatdir, chunki u ularni keng ko'lamli apparat konfiguratsiyalarida ishlatish va har xil turdagi kompyuterlar o'rtasida osonlik bilan o'tkazish imkonini beradi. Bu, ayniqsa, har xil turdagi qo'shimcha qurilmalar qo'llaniladigan yoki vaqt o'tishi bilan jihozlar almashtirilishi yoki yangilanishi mumkin bo'lgan muhitlarda foydali bo'lishi mumkin.

Mashinaga bog'liq kodni optimallashtirish - bu ma'lum bir turdagi apparat uchun kodni optimallashtirishga qaratilgan kodni optimallashtirish turi. Ushbu turdagi optimallashtirish odatda montaj tili bilan amalga oshiriladi, bu ma'lum bir turdagi apparat bilan foydalanish uchun mo'ljallangan past darajadagi dasturlash tilidir. Uskunaning o'ziga xos xususiyatlaridan foydalangan holda, kodni tezroq va samaraliroq ishlash uchun optimallashtirish mumkin. Mashinaga bog'liq kodni optimallashtirishning afzalligi shundaki, u sezilarli samaradorlikni ta'minlashi mumkin. Buning sababi shundaki, kod apparatning o'ziga xos xususiyatlaridan foydalanish uchun maxsus ishlab chiqilgan. Salbiy tomoni shundaki, kod faqat ma'lum bir uskunada ishlashi mumkin, ya'ni uni jiddiy qayta ishlamasdan boshqa turdagi uskunaga o'tkazib bo'lmaydi.

Mashinaga bog'liqmas kod optimallashtirish - bu har qanday turdagi apparat uchun kodni optimallashtirishga qaratilgan kodni optimallashtirish turi. Ushbu turdagi optimallashtirish odatda C, C++ va Java kabi yuqori darajadagi tillar bilan amalga oshiriladi. Ushbu tillarning imkoniyatlaridan foydalanib, har qanday turdagi apparat

uchun optimallashtirilgan kod yozish mumkin. Mashinadan mustaqil kodni optimallashtirishning afzalligi shundaki, kod muhim qayta ishlanmasdan har qanday turdagi uskunaga o'tkazilishi mumkin. Bu shuni anglatadiki, kodni bir platformadan boshqasiga osongina ko'chirish mumkin, bu esa uni bir nechta platformalarda ishlashga imkon beradi. Salbiy tomoni shundaki, u mashinaga bog'liq kodni optimallashtirish kabi samarali emas, chunki u har qanday muayyan uskunaning o'ziga xos xususiyatlariga moslashtirilmagan.

Quyidagi muammoga yechim topamiz:

Dasturchilar tomonidan yozilgan arifmetik ifodalarda ko‘p hollarda bir xil ifoda bir nechta marta qayta ishlatiladi. Ammo ba’zi kompilyatorlar yoki dasturiy muhitlar bu takrorlanuvchi arifmetik ifodalarni avtomatik aniqlab, optimallashtirishni bajarmaydi, natijada:

1. a)Hisoblashlar soni ortadi,

b)Ijro muddati cho‘ziladi,

c)Resurslar behuda sarflanadi.

2. Hayotiy misol (real kod parchasi):

```
double a = 5.2, b = 7.8, c = 1.1;
double result = Math.Pow((a + b + c), 2) + Math.Sqrt(a + b + c) + Math.Log(a + b + c);
```

Bu yerda (a + b + c) ifodasi uch marta hisoblanmoqda.

3. Bu muammoga qanday yechim topish mumkin?

Har bir arifmetik amal mashina resurslaridan foydalanadi (CPU, RAM).

Real vaqtli yoki katta hajmli hisoblashlarda bu foydalanuvchi interfeysi sekinlashuvi yoki samaradorlikning pasayishiga olib keladi.

Ba’zi kompilyatorlar CSE (Common Subexpression Elimination) texnikasini har doim qo‘llamaydi, ayniqsa bu ifoda funksiya ichida yoki shartli bloklarda bo‘lsa.

4. Muammoning yechimi – Mashinaga bog‘liqmas optimallashtirish:

Kompilyatordan qat'i nazar, dasturchi o‘zi ifodani soddalashtiradi:

```
double t = a + b + c;
double result = Math.Pow(t, 2) + Math.Sqrt(t) + Math.Log(t);
```

5. Tahlil:

Ko‘rsatkich	**Avvalgi holat**	**Optimallashtirilgan holat**
(a + b + c) hisoblashlar soni	3 marta	1 marta
Resurs sarfi	Ko‘p	Kamroq

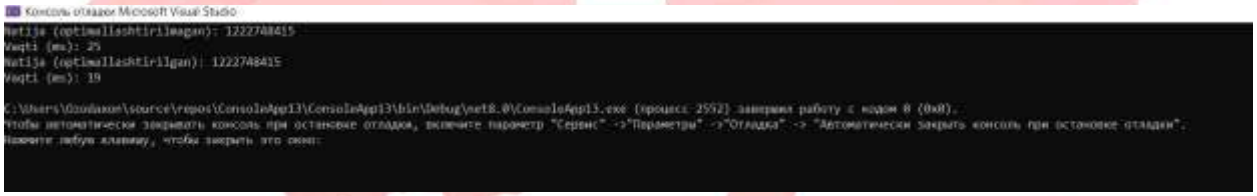
Ko‘rsatkich	Avvalgi holat	Optimallashtirilgan holat
Kod o‘qilishi	Chalkashroq	Ancha tushunarli
Mashinaga bog‘liqmi?	Ha (kompilyatorga bog‘liq)	Yo‘q (kodda hal qilingan)

Bu muammo kichik ko‘rinsa-da, keng miqyosdagi dasturlar, ilmiy hisob-kitoblar, real vaqt tizimlari va o‘yin dvijoklari uchun sezilarli farq qiladi. Mashinaga bog‘liqmas optimallashtirish orqali bunday ifodalarni dastur darajasida oldindan aniqlab, bitta o‘zgaruvchiga biriktirish orqali umumiy samaradorlik oshiriladi.

Quyidagi kod orqali ham bu masalani korib chiqishimiz mumkin:

```
using System;
using System.Diagnostics;

namespace ArithmeticOptimization
{
class Program
{
  static void Main(string[] args)
{
 int a = 3, b = 5, c = 2;

  Stopwatch sw = new Stopwatch();

  // Optimallashtirilmagan
   sw.Start();
  int resultBefore = ComplexCalculation(a) + ComplexCalculation(b) + ComplexCalculation(a + b) - ComplexCalculation(c);
      sw.Stop();
   Console.WriteLine("Natija (optimallashtirilmagan): " + resultBefore);
   Console.WriteLine("Vaqti (ms): " + sw.ElapsedMilliseconds);

    // Optimallashtirilgan
    sw.Restart();
    int calcA = ComplexCalculation(a);
```

```
            int calcB = ComplexCalculation(b);
            int calcAB = ComplexCalculation(a + b);
            int calcC = ComplexCalculation(c);
            int resultAfter = calcA + calcB + calcAB - calcC;
            sw.Stop();
            Console.WriteLine("Natija (optimallashtirilgan): " + resultAfter);
            Console.WriteLine("Vaqti (ms): " + sw.ElapsedMilliseconds);
        }

        static int ComplexCalculation(int x)
        {
            int sum = 0;
            for (int i = 1; i <= 1000000; i++) // Kattaroq iteratsiya, farq sezilsin
            {
                sum += (x * i + x % (i + 1)) / (i % 5 + 1);
            }
            return sum;
        }
    }
}
```



Mashinaga bog'liq kodni optimallashtirish	**Mashinadan mustaqil kodni optimallashtirish**
Mashinaga bog'liq kodni optimallashtirish ma'lum turdagi kompyuter yoki apparat platformasi uchun kodni optimallashtirishni anglatadi	Mashinadan mustaqil kodni optimallashtirish har qanday turdagi uskunada foydalanish uchun kodni optimallashtirishni anglatadi.

Mashinaga bog'liq kodni optimallashtirish	Mashinadan mustaqil kodni optimallashtirish
Mashinaga bog'liq kodni optimallashtirishning maqsadi ma'lum bir apparat platformasining o'ziga xos xususiyatlari yoki xususiyatlaridan foydalanishdir.	Mashinadan mustaqil kodni optimallashtirishning maqsadi kodni barcha apparat platformalarida iloji boricha samaraliroq qilishdir.
Mashinaga bog'liq kodni optimallashtirish ma'lum bir apparat platformasi bilan cheklangan.	Mashinadan mustaqil kod optimallashtirish har qanday apparat platformasida qo'llaniladi.
Mashinaga bog'liq kodni optimallashtirish boshqa apparat platformalari bilan mos kelmaydigan kodga olib kelishi mumkin	Mashinadan mustaqil kodni optimallashtirish barcha apparat platformalariga mos keladigan kod ishlab chiqaradi
Mashinaga bog'liq kodni optimallashtirish kodni boshqa apparat platformalariga o'tkazishni qiyinlashtirishi mumkin. Mashinadan mustaqil kod optimallashtirish maqsadlari	Mashinadan mustaqil kodni optimallashtirish kodni boshqa platformalarga o'tkazishni osonlashtiradi.
Mashinaga bog'liq kodni optimallashtirish, turli apparat platformalari uchun optimallashtirilgan kodni saqlash uchun ko'proq texnik xizmat ko'rsatish va yangilanishlarni talab qilishi mumkin.	Mashinadan mustaqil kodni optimallashtirish kamroq yangilanishlarni talab qiladi va uni saqlash osonroq.
Mashinaga bog'liq kodni optimallashtirishni ishlab chiqish ko'proq vaqt talab qilishi mumkin, chunki u apparat platformasi bo'yicha	Mashinadan mustaqil kodni optimallashtirish tezroq ishlab chiqilishi mumkin.

Mashinaga bog'liq kodni optimallashtirish	Mashinadan mustaqil kodni optimallashtirish
maxsus bilimlarni talab qiladi va bir nechta platformalarda sinovni talab qilishi mumkin.	
Mashinaga bog'liq kodni optimallashtirish muayyan apparat platformasida yaxshi ishlashga olib kelishi mumkin.	Mashinadan mustaqil kodni optimallashtirish barcha platformalarda yanada barqaror ishlashga olib kelishi mumkin.
Mashinaga bog'liq kodni optimallashtirish yanada murakkab kodni talab qilishi mumkin, chunki u apparat platformasining o'ziga xos xususiyatlarini hisobga olishi kerak.	Mashinadan mustaqil kodni optimallashtirish oddiyroq, umumiy koddan foydalanishi mumkin.
Mashinaga bog'liq kodni optimallashtirish boshqa apparat platformalarida qayta ishlatilmasligi mumkin.	Mashinadan mustaqil kodni optimallashtirish har qanday platformada osongina qayta ishlatilishi mumkin.

Xulosa

Men ushbu maqolamda arifmetik ifodalarni mashinaga bog’liqmas optimallashtirishning ahamiyati va uning dasturiy ta'minotni samarali va portativ qilishdagi rolini ko'rib chiqishga harakat qildim. Ushbu optimallashtirish turi, mashinaning arxitekturasidan mustaqil bo‘lib, barcha turdagi platformalarda ishlashni ta'minlaydi. Arifmetik ifodalarning optimallashtirilishi nafaqat kodni qisqartiradi, balki uni tezroq bajarilishiga imkon yaratadi, shu bilan birga tizim resurslaridan yanada samarali foydalanishni ta'minlaydi.

Mashinaga bog’liqmas optimallashtirishning asosiy maqsadi — arifmetik amallarni soddalashtirish yoki o‘zgartirish orqali kodning umumiy samaradorligini oshirishdir. Bu usul dasturiy ta'minotning portativligini ta'minlaydi va bitta kodni turli kompyuter tizimlarida va operatsion tizimlarda muammosiz ishlashiga imkon yaratadi. Ushbu

jarayonda ishlatiladigan texnikalar, masalan, doimiylarni oldindan hisoblash, takrorlanuvchi ifodalarni yo‘qotish yoki arifmetik amallarni soddalashtirish, kodning tezligini oshirishga yordam beradi.

Shuningdek, kompilyatorlar bu optimallashtirishni avtomatik ravishda amalga oshiradi, bu esa dasturchilarga ko‘p qo‘shimcha ishni yengillashtiradi. Kompilyatorlar arifmetik ifodalarni tahlil qiladi va ularni optimallashtiradi, natijada dastur ishlashining samaradorligi oshadi. Misol uchun, x * 1 + 0 kabi oddiy ifodalar avtomatik ravishda soddalashtiriladi, bu esa tizim resurslarining tejanishiga olib keladi.

Xulosa qilib aytganda, arifmetik ifodalarni mashinaga bog‘liqmas optimallashtirish dasturiy ta'minotni samarali, barqaror va portativ holatda yaratishda muhim vosita hisoblanadi. Ushbu yondashuv orqali kod soddalashtiriladi, ortiqcha va takroriy hisob-kitoblar yo‘qotiladi, bu esa bajarilish tezligini oshiradi va tizim resurslaridan oqilona foydalanishga imkon yaratadi. Eng muhimi, bunday optimallashtirish usullari har qanday platforma va arxitekturada barqaror ishlaydigan, ko‘p tizimli dasturiy echimlar ishlab chiqishga zamin hozirlaydi.

Kompilyatorlar yordamida ushbu optimallashtirish jarayonini avtomatlashtirish dasturchilar uchun katta yengillik yaratadi. Natijada, dastur faqat samarali ishlash bilan cheklanmay, balki unga texnik xizmat ko‘rsatish, kengaytirish va turli muhitlarga ko‘chirish jarayonlari ham ancha soddalashadi.

Shuningdek, optimallashtirilgan kod energetik samaradorlikka ham xizmat qiladi — bu esa mobil qurilmalar va kam resursli tizimlar uchun juda dolzarbdir. Shu sababli, arifmetik ifodalarni mashinaga bog‘liqmas tarzda optimallashtirish zamonaviy dasturiy ta'minot ishlab chiqish jarayonining ajralmas va strategik jihatdan muhim bosqichidir.

Foydalanilgan adabiyotlar

1. B.Umarov., M.Umarova. THE PROBLEM OF APPROXIMATING SIGNALS BASED ON MODELING OF WAVELET - HAAR TRANSFORMATION. - 2020. - C. 502-506.

2. Azizovich.U B INNOVATSION TEXNOLOGIYALAR ORQALI O'QITUVCHILAR KOMETANSIYATINI SHAKLLANTIRISH TASOSIYLARI. Finlyandiya xalqaro ta'lim ilmiy jurnali. Ijtimoiy va gumanitar fanlar. 2023;11(5):823-8

3. Umarov, B., G'ulomjonova, S. (2024). BULUT TEXNOLOGIYASI VA ULARDAN FOYDALANISH. Zamonaviy ta'limda innovatsion tadqiqotlar, 2(7), 12-14. Qaytadan olindi

4. Umarov A.K., Mahkamov A.A. – *Dasturlash tillari va algoritmlar asoslari*. Toshkent: TDPU, 2019.

5. B.Umarov, V.Jo'rayev "Kompyuter tarmoqlari", 2024
6. Islomov O.O., To'rayev A.A. – *Algoritmlar va ma'lumotlar tuzilmasi*. Toshkent: Iqtisodiyot, 2018.
7. Umarov A.K. – *Algoritmlarni tahlil qilish va samarali dasturlash usullari*. Toshkent: Innovatsiya, 2021.
8. Karimov B.B. – *Dasturlash asoslari (C tilida)*. Toshkent: TUIT, 2020.
9. Rahmonov Sh.A. – *Kompyuter tizimlari va dasturlash*. Toshkent: Fan va texnologiya, 2017.
10. Shodiyev Sh.R., Mardonov M.M. – *Dasturlash tillari (Python, C++)*. Toshkent: TDYuU, 2022.
11. Aho A.V., Lam M.S., Sethi R., Ullman J.D. – *Compilers: Principles, Techniques, and Tools*. 2nd Edition, 2006.
12. Kennedy K., Allen J.R. – *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*. Elsevier, 2002.
13. Grigoryev D.V., Nikolayev A.V. – *Kompilyatorlar tuzilishi va ishlash prinsipi*. Moskva: BINOM, 2017.
1. https://arxiv.uz/uz/
2. www.library.ziyonet.uz