ISSN 2181-3187

MASHINAGA BOG'LIQ BO'LMAGAN ARIFMETIK IFODALARNI OPTIMALLASHTIRISH

Tojimamatov Israiljon Nurmamatovich
Farg`ona davlat universiteti
israiltojimamatov@gmail.com
Ne`matova Shohsanam Nodirbek qizi
Farg'ona davlat universiteti talabasi
nematovashohsanam5@gmail.com

Anotatsiya: Ushbu maqola dasturiy ta'minotning ishlashini yaxshilashga qaratilgan mashinaga bog'liq bo'lmagan optimallashtirish usullaridan biri bo'lgan arifmetik ifodalarni optimallashtirish masalasini ko'rib chiqadi. Maqolada doimiy katlama, algebraik soddalashtirish, umumiy subifodalarni yo'q qilish, kodni ko'chirish, kuchni kamaytirish va o'lik kodni yo'q qilish kabi asosiy optimallashtirish texnikalari tahlil qilinadi. Ushbu usullarning dastur kodining samaradorligini oshirishdagi roli va ularning qo'llanilish mexanizmlari misollar bilan yoritilgan. Maqola so'ngida mashinaga bog'liq bo'lmagan optimallashtirishning dasturiy ta'minotni ishlab chiqish jarayonidagi ahamiyati va istiqbollari muhokama qilinadi.

Kalit so‘zlar: Mashinaga bog'liq bo'lmagan optimallashtirish, Arifmetik ifodalarni optimallashtirish, Kompilyator optimallashtirish, Doimiy katlama, Algebraik soddalashtirish, Umumiy subifodalar, Kodni ko'chirish, Kuchni kamaytirish, O'lik kodni yo'q qilish, Dasturiy ta'minotning ishlashi.

Аннотация: Данная статья рассматривает проблему оптимизации арифметических выражений, которая является одним из машинно-независимых методов оптимизации, направленных на повышение производительности программного обеспечения. В статье анализируются ключевые методы оптимизации, такие как свертка констант, алгебраическое упрощение, устранение общих подвыражений, перемещение кода, понижение мощности и устранение мертвого кода. Роль этих методов в повышении эффективности

ISSN 2181-3187

программного кода и механизмы их применения иллюстрируются примерами. В заключение статьи обсуждаются значение и перспективы машинно-независимой оптимизации в процессе разработки программного обеспечения.

Ключевые слова: Машинно-независимая оптимизация, Оптимизация арифметических выражений, Оптимизация компилятора, Свёртка констант, Алгебраическое упрощение, Общие подвыражения, Перемещение кода, Понижение мощности, Устранение мертвого кода, Производительность программного обеспечения.

Annotation: This article examines the issue of optimizing arithmetic expressions, which is one of the machine-independent optimization methods aimed at improving software performance. The paper analyzes key optimization techniques such as constant folding, algebraic simplification, common subexpression elimination, code motion, strength reduction, and dead code elimination. The role of these methods in enhancing the efficiency of program code and their application mechanisms are illustrated with examples. The article concludes by discussing the significance and prospects of machine-independent optimization in the software development process.

Keywords: Machine-independent optimization, Arithmetic expression optimization, Compiler optimization, Constant folding, Algebraic simplification, Common subexpressions, Code motion, Strength reduction, Dead code elimination, Software performance.

Kirish: Dasturiy ta'minotning ishlashini yaxshilashga intilish nafaqat yakuniy foydalanuvchi tajribasini oshirish, balki resurslardan (masalan, protsessor vaqti, xotira, energiya) yanada oqilona foydalanish zaruriyatidan ham kelib chiqadi. Zamonaviy dasturlar tobora murakkablashib borar ekan, optimallashtirish texnikalarining ahamiyati yanada ortib bormoqda. Kompilyatsiya jarayonida qo'llaniladigan optimallashtirishlar, xususan mashinaga bog'liq bo'lmagan optimallashtirishlar, dastur kodini apparatning o'ziga xos xususiyatlariga bog'lamasdan turib, uning fundamental samaradorligini oshirishga qaratilgan. Bu esa, optimallashtirilgan kodning turli xil platformalarda yaxshi ishlashini ta'minlaydi va dastur ishlab chiqish jarayonini

ISSN 2181-3187

soddalashtiradi. Arifmetik ifodalarni optimallashtirish ushbu jarayonning muhim bir qismi bo'lib, hisoblashlarni soddalashtirish, ortiqcha operatsiyalarni bartaraf etish va umumiy bajarilish tezligini oshirish orqali dastur samaradorligiga bevosita ta'sir ko'rsatadi.

Mashinaga bog'liq bo'lmagan optimallashtirish - bu kompilyatsiya jarayonining muhim bosqichi bo'lib, u oraliq kod (masalan, uch adresli kod) yoki dastlabki manba kodining o'zini yaxshilashga yo'naltirilgan. Ushbu bosqichning o'ziga xosligi shundaki, u ma'lum bir mikroprotsessorning arxitekturasi, uning instruktsiyalari to'plami, registrlar soni, kesh xotirasi hajmi yoki boshqa apparat xususiyatlari haqida hech qanday **aniq taxminlarga asoslanmaydi**. Buning o'rniga, mashinaga bog'liq bo'lmagan optimallashtirishlar dastur mantig'ini **o'zgartirmasdan**, balki uni yanada **samaraliroq** bajarilishini ta'minlash uchun turli xil **umumiy tahlil va o'zgartirish texnikalaridan** foydalanadi. Bularga algebraik qonunlarni qo'llash orqali ifodalarni soddalashtirish, dastur kodidagi takroriy yoki ortiqcha hisoblashlarni aniqlash va bartaraf etish, shuningdek, boshqaruv oqimini tahlil qilish va kodni qayta tashkil etish kabi usullar kiradi. Ushbu optimallashtirishlar natijasida hosil bo'lgan yaxshilangan oraliq kod keyingi bosqichda, mashinaga bog'liq optimallashtirish va kodni generatsiya qilish bosqichlarida, ma'lum bir maqsadli mashina arxitekturasining imkoniyatlaridan to'liq foydalanish uchun asos yaratadi.

Qo'shimcha jihatlar:

- **Kompilyatsiya bosqichi:** Mashinaga bog'liq bo'lmagan optimallashtirishning kompilyatsiya jarayonidagi o'rni aniqlashtiriladi.
- **Oraliq kod:** Optimallashtirish ko'pincha oraliq kod ustida amalga oshirilishi ta'kidlanadi.
- **Apparatga bog'liq emaslikning sabablari:** Nega bu bosqichda apparat haqida taxmin qilinmasligi tushuntiriladi (umumiy samaradorlikka qaratilganlik).

ISSN 2181-3187

- **Foydalaniladigan texnikalarning turlari:** Algebraik qonunlar, kodni tahlil qilish, boshqaruv oqimini tahlil qilish va kodni qayta tashkil etish kabi kengroq tushunchalar qo'shiladi.
- **Maqsadi:** Optimallashtirishning asosiy maqsadi dastur mantig'ini saqlab qolgan holda samaradorlikni oshirish ekanligi yana bir bor ta'kidlanadi.
- **Keyingi bosqichga zamin yaratishi:** Mashinaga bog'liq bo'lmagan optimallashtirishning keyingi kompilyatsiya bosqichlari uchun muhim asos bo'lishi ko'rsatiladi.

Arifmetik Ifodalarni Optimallashtirish Usullari

Arifmetik ifodalarni optimallashtirish mashinaga bog'liq bo'lmagan optimallashtirishning muhim bir qismidir. Ushbu optimallashtirishlar hisoblash jarayonini soddalashtirish, takroriy hisoblashlarni bartaraf etish va umumiy ishlashni yaxshilashga qaratilgan. Quyida eng ko'p qo'llaniladigan usullar keltirilgan:

1. **Doimiy Katlama (Constant Folding)**

Doimiy katlama - bu kompilyator tomonidan kompilyatsiya vaqtida qiymati ma'lum bo'lgan ifodalarni hisoblash jarayonidir. Agar ifodada faqat doimiylar ishtirok etsa, kompilyator ushbu ifodani hisoblab, natijani kodga kiritadi.

Misol:

```
x = 2 + 3 * 5;
```

Optimallashtirilgandan so'ng:

```
x = 17;
```

Doimiy katlama dastur bajarilishi vaqtida ortiqcha hisoblashlarning oldini oladi va kodni yanada tezroq qiladi.

2. **Algebraik Soddalashtirish (Algebraic Simplification)**

Algebraik soddalashtirish algebraik qonunlardan foydalanib ifodalarni soddalashtirishni o'z ichiga oladi. Bu ortiqcha operatsiyalarni olib tashlash va ifodalarni yanada samaraliroq shaklga keltirishga yordam beradi.

Misollar:

- $x * 1 \rightarrow x$

ISSN 2181-3187

- x + 0 → x
- x - 0 → x
- x * 0 → 0
- (a + b) - b → a
- a * (b + c) → a * b + a * c (distributivlik)
- 2 * x + 3 * x → 5 * x (yig'ish)

Algebraik soddalashtirish orqali kompilyator murakkab ifodalarni oddiyroq va tezroq bajariladigan shakllarga aylantirishi mumkin.

3. **Umumiy Subifodalarni Yo'q qilish (Common Subexpression Elimination)**

Agar bir xil subifoda dastur kodining bir nechta joyida takrorlansa, kompilyator ushbu subifodani bir marta hisoblab, natijasini vaqtinchalik o'zgaruvchida saqlashi va keyingi o'rinlarda ushbu o'zgaruvchidan foydalanishi mumkin. Bu takroriy hisoblashlarning oldini oladi.

Misol:

```
a = b * c + d;
e = b * c * f;
```

Optimallashtirilgandan so'ng:

```
temp = b * c;
a = temp + d;
e = temp * f;
```

Umumiy subifodalarni yo'q qilish dastur ishlashini sezilarli darajada yaxshilashi mumkin, ayniqsa murakkab hisoblashlar ko'p marta takrorlanganda.

4. **Kodni Ko'chirish (Code Motion)**

Tsikl ichida o'zgarmaydigan (invariant) ifodalar mavjud bo'lsa, ularni tsikldan tashqariga ko'chirish orqali har bir iteratsiyada qayta hisoblashning oldini olish mumkin.

Misol:

```
for i = 0 to 100 do
```

ISSN 2181-3187

```
    x = y + z; // y va z tsikl ichida o'zgarmaydi
    a[i] = a[i] * x;
end for;
```

Optimallashtirilgandan so'ng:

```
x = y + z;
for i = 0 to 100 do
    a[i] = a[i] * x;
end for;
```

Kodni ko'chirish tsikl ichidagi ortiqcha hisoblashlarni kamaytiradi va tsiklning umumiy bajarilish vaqtini tezlashtiradi.

5. **Kuchni Kamaytirish (Strength Reduction)**

Kuchni kamaytirish - bu qimmatroq arifmetik operatsiyalarni arzonroq operatsiyalarga almashtirish texnikasidir. Masalan, ko'paytirishni qo'shishga yoki darajaga ko'tarishni ko'paytirishga almashtirish mumkin.

Misollar:

- x * 2 → x + x yoki x << 1 (agar x butun son bo'lsa)
- x ^ 2 → x * x
- Tsikl ichidagi indekslarni hisoblashda ko'paytirish o'rniga qo'shishdan foydalanish (masalan, massiv elementlariga murojaat qilishda).

Kuchni kamaytirish operatsiyalarning bajarilish vaqtini qisqartirish orqali dastur ishlashini yaxshilaydi.

6. **O'lik Kodni Yo'q qilish (Dead Code Elimination)**

O'lik kod - bu dastur natijasiga ta'sir qilmaydigan koddir. Bunga hech qachon ishlatilmaydigan o'zgaruvchilar, hech qachon bajarilmaydigan kod bloklari yoki natijasi hech qachon ishlatilmaydigan hisoblashlar kiradi. Kompilyator bunday kodni aniqlab, olib tashlashi mumkin.

Misol:

```
x = y + z; // x dan keyin hech qachon foydalanilmaydi
a = 5;
```

ISSN 2181-3187

```
if (false) {
    b = c * d; // Bu kod hech qachon bajarilmaydi
}
```

Optimallashtirilgandan so'ng:

```
a = 5;
```

O'lik kodni yo'q qilish dastur hajmini kamaytiradi va kompilyatsiya jarayonini tezlashtiradi.

Xulosa: Mashinaga bog'liq bo'lmagan arifmetik ifodalarni optimallashtirish kompilyatorning muhim bosqichlaridan biri bo'lib, dasturiy ta'minotning ishlashini sezilarli darajada yaxshilashga yordam beradi. Doimiy katlama, algebraik soddalashtirish, umumiy subifodalarni yo'q qilish, kodni ko'chirish, kuchni kamaytirish va o'lik kodni yo'q qilish kabi usullar orqali kompilyatorlar dastur kodini yanada samaraliroq va tezroq bajariladigan shaklga keltirishi mumkin. Ushbu optimallashtirishlar ma'lum bir apparat arxitekturasiga bog'liq bo'lmaganligi sababli, ular turli xil platformalarda dasturiy ta'minotning umumiy ishlashini yaxshilash uchun qo'llanilishi mumkin. Zamonaviy kompilyatorlar ushbu va boshqa ko'plab optimallashtirish texnikalarini avtomatik ravishda qo'llab, dasturchilarga yanada samarali va tezkor dasturlar yaratishga imkon beradi.

Foydalanilgan adabiyotlar:

1. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006).
2. Muchnick, S. S. (1997).
3. Cooper, K., & Torczon, L. (2011).
4. Grune, D., van Reeuwijk, K., Bal, H. E., Jacobs, C. J. H., & Langendoen, K. (2012).
5. Karimberdiyevich, O. M., & Abdulaziz o‘g’li, Y. M. (2024). SUN’IY INTELLEKTNING AFZALLIKLARI VA KAMCHILIKLARI. IZLANUVCHI, 1(1), 75-85.

6. Karimberdiyevich, O. M., & Abdulaziz o‘g‘li, Y. M. (2024). NEYRO KOMPYUTERLAR. YANGI O ‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI, 1(5), 19-27.

7. Karimberdiyevich, O. M., & Abdulaziz o‘g‘li, Y. M. (2024). K-YAQIN QO'SHNI ALGORITMI. IZLANUVCHI, 1(1), 122-124.

8. Abdulaziz o'g'li, Y. M. (2025). WPFDA ANIMATSIYA YARATISHNI QO’LLANISHI. MODERN PROBLEMS IN EDUCATION AND THEIR SCIENTIFIC SOLUTIONS, 1(4), 172-175.

9. Abdulaziz o’g’li, Y. M. (2025). MOLIYA VA HISOB–KITOB ILOVALARIDA WPF BILAN ISHLASH. MODERN PROBLEMS IN EDUCATION AND THEIR SCIENTIFIC SOLUTIONS, 1(4), 189-193.

10. Karimberdiyevich, O. M. (2024). NEYROEMULYATORLAR VA ULARNING QO'LLANILISHI. YANGI O ‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI, 1(5), 82-89.

11. Abdulaziz ogli, Y. M. (2025). WPF DA IKKI O’LCHOVLI VA UCH O’LCHOVLI GRAFIKALAR BILAN ISHLASHNING HAYOTGA TATBIQLARI. MODERN PROBLEMS IN EDUCATION AND THEIR SCIENTIFIC SOLUTIONS, 1(4), 176-179.

12. Karimberdiyevich, O. M., & Abdulaziz o’g’li, Y. M. (2024). MASHINANI O ‘RGANISHDA TASNIFLASH VA REGRESSIYA. IZLANUVCHI, 1(1), 114-121.

13. Karimberdiyevich, O. M., & Abdulaziz o‘g‘li, Y. M. (2024). PIVOT JADVALI YARATISH VA TAHRIRLASH. YANGI O ‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI, 1(5), 28-30.

14. Karimberdiyevich, O. M., & Abdulaziz o’g’li, Y. M. (2024). MASHINANI O ‘RGANISHDA TASNIFLASH VA REGRESSIYA. IZLANUVCHI, 1(1), 114-121.

15. Karimberdiyevich, O. M. (2024). FORMAL GRAMMATIKA VA SEMANTIK TO’R. IZLANUVCHI, 1(1), 94-99.

16. Karimberdiyevich, O. M., & Abdulaziz o‘g’li, Y. M. (2024). PROGNOZLASH VA VIZUALIZATSIYA. TA’LIM, TARBIYA VA INNOVATSIYALAR JURNALI, 1(1), 124-132.

17. Karimberdiyevich, O. M. (2024). O’ZBEKISTON RESPUBLIKASIDA BIG DATA NI RIVOJLANISH TENDENSIYALARI. TA’LIM, TARBIYA VA INNOVATSIYALAR JURNALI, 1(1), 147-151.

18. Karimberdiyevich, O. M., & Abdulaziz o‘g‘li, Y. M. (2024). ICHKI MUAMMOLARNI TUSHUNISH. TA’LIM, TARBIYA VA INNOVATSIYALAR JURNALI, 1(1), 98-104.

19. Karimberdiyevich, O. M., Abdulaziz o‘g‘li, Y. M., & Hokimjon o’g, I. M. R. (2024). EVALUTSION DASTURLASH. GENETIK ALGORITM. YANGI O ‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI, 1(4), 519-522.

20. "Optimizing Compilers for Modern Architectures"

21. "High Performance Browser Networking"

22. "Performance Optimization Techniques in C# and .NET"

23. "Computer Systems: A Programmer’s Perspective"

24. "Modern Compiler Implementation in C / Java / ML"

25. **"Software Optimization Guide"** (Intel Developer Manual)