DOI: https://doi.org/10.33216/2222-3428-2019-17-9

УДК: 681.3.07

Недзельський Д.О., Сафонова С.О.

ПРО РЕАЛЬНУ ПРОДУКТИВНІСТЬ ТА ЕФЕКТИВНІСТЬ ЯДЕР СУЧАСНИХ ПРОЦЕСОРІВ

У статті розглянуті причини невідповідності пікової та реальної продуктивності комп'ютерів, досліджені діапазони зміни реальної продуктивності та реальної ефективності ядер сучасних процесорів за наявності у виконуваних програмах таких «перешкод» для конвеєра ядра процесора, як інформаційні залежності, операції редукції, команди переходів. Розроблена модель ядра процесора. Незбалансованість в роботі ядра при виконанні реальних програм обумовлюється як особливостями програм, так і структурними рішеннями ядра процесора і комп'ютера в цілому. У статті аналіз продуктивності кожної фази моделі ядра виконується при роботі ядра в найбільш сприятливих умовах для структур ядер. Вплив «перешкод» у вигляді команд переходів, інформаційних залежностей між командами, операцій редукції на реальну продуктивність і ефективність ядра процесора розглянуто на прикладах програм «Множення матриць» і «Рішення диференціальних рівнянь в приватних похідних методом сіток». Реальна продуктивність і ефективність ядра процесора при виконанні ядра програми «Множення матриць» в ідеальних умовах складають не більше 1 команди за такт, ефективність - менше 0.25. Час роботи підсистеми виконання команд ядра процесора істотно залежить як від властивостей виконуваних програм (від кількості інформаційно залежних команд, наявності операцій редукції), так і від часу виконання команд, насамперед команд звернення до оперативної пам'яті. При виконанні багатьох реальних програм продуктивність ядра процесора визначається продуктивністю підсистеми виконання команд. Головні причини цього - інформаційні залежності між командами програми і операції редукції. Реальна продуктивність ядра процесора може бути меншою теоретичної продуктивності у багато разів.

Ключові слова*: процесор, ядро, конвеєр команд, продуктивність, ефективність, перехід, підсистема виконання команд.*

Вступ. Розрізняють теоретичну і реальну продуктивності. На практиці важлива реальна продуктивність Rmax, досягнута при виконанні конкретної програми, яка є відношенням об'єму виконаної роботи (числа виконаних операцій або кількості виконаних команд) програми до часу її виконання. Відношення E = Rmax/Rpeak визначає ефективність роботи комп'ютера.

У літературі і в повсякденній практиці часто обговорюються дані про продуктивність як комп'ютерів в цілому, так і окремих їх підсистем (наприклад, ядер, оперативної пам'яті, кеш-пам'ятей і тому подібне). При цьому оперують, як правило, показниками пікової продуктивності. Показники пікової продуктивності корисні, передусім, для порівняння комп'ютерів (та їх підсистем) різних виробників або комп'ютерів одного виробника, але різних років випуску. Проте конкретного користувача цікавить не теоретична продуктивність, а реальна продуктивність його комп'ютера при виконанні конкретних програм.

При порівнянні реальної продуктивності з теоретичною продуктивністю виявляється, що реальна продуктивність комп'ютера дуже далека від теоретичної продуктивності. Наприклад, в [1] приведені дані про ефективність (відношення Rmax/Rpeak) ряду нескладних тестів для ядер деяких процесорів фірми Intel, які знаходяться в інтервалі від 20 до декількох відсотків.

За допомогою програмних засобів фірми Intel було проведено дослідження ефективності роботи одного ядра процесора Clowertown 2,66 ГГц при виконанні завдання комп'ютерного дизайну ліків. Реальна продуктивність ядра процесора виявилася приблизно 400 MFLOPS при піковій продуктивності 10,64 GFLOPS. Виходить, що коефіцієнт корисної дії (ККД) роботи ядра процесора при виконанні цієї програми близько 4%, що менше, ніж ККД паровоза.

В [2] стверджується, що користуватися піковою продуктивністю не доречно.

Можна припустити, що дані [1] застаріли, і що сучасніші ядра процесорів виконують розглянуті тести значно ефективніше. Проте уважний розгляд структур найсучасніших ядер (наприклад, структур ядер фірми Intel [3-6], ZEN 2 - фірми AMD [7-8]) показує, що принципово вони кардинально не відрізняються від попередніх структур ядер. Основні принципові рішення в структурах нових ядер ті ж, що і в структурах попередніх ядер.

Деякі питання ефективності ядер сучасних процесорів досліджувалися в роботах [9-11]. Зокрема, в [9] досліджувалися питання ефективності одноядерних суперскалярних обчислювальних систем. У [10] досліджувалася ефективність структурних методів компенсації впливу команд переходів на продуктивність конвеєрних ядер процесорів. У [11] досліджувалася ефективність підсистеми генерації команд в ядрах сучасних процесорів.

У літературі правильно відзначається, що однією з причин істотного зменшення реальної продуктивності ядер процесорів (і, відповідно, усього комп'ютера) від їх теоретичної продуктивності є вплив оперативної пам'яті, особливо при значній кількості звернень до неї. Проте відсутні дослідження впливу на реальну продуктивність ядер процесорів таких особливостей виконуваних програм як інформаційні залежності, операції редукції.

Метою статті є дослідження діапазонів зміни реальної продуктивності та реальної ефективності ядер сучасних процесорів за наявності у виконуваних програмах таких «перешкод» для конвеєра ядра процесора, як інформаційні залежності, операції редукції, команди переходів.

Основний зміст роботи.

Розробка моделі ядра процесора

Довжина реальних конвеєрів команд у високопродуктивних ядрах сучасних процесорів складає не менше 10 етапів. Ряд перших етапів конвеєра ядра виконує функції по читанню команд з підсистеми пам'яті, буферизації їх, дешифрації та підготовці до виконання. Подальші етапи конвеєра команд ядра виконують підготовлені (дешифровані) команди.

З точки зору дослідження продуктивності (ефективності) роботи ядра процесора доцільно представити конвеєр ядра у вигляді 2 фаз (етапів) - фази підготовки команд і фази виконання підготовлених (декодованих) команд з буфером між ними (рис. 1).

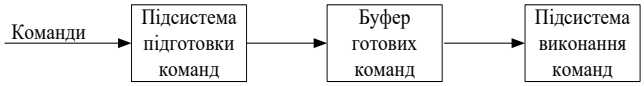

Рис. 1 - Структура моделі ядра процесора

Залежно від співвідношення продуктивностей фаз про продуктивність моделі ядра можна судити по продуктивності будь-якої фази - якщо продуктивності фаз моделі рівні, або по продуктивності повільнішої фази, якщо продуктивності фаз моделі не рівні. Пропускна спроможність повільнішої фази визначається часом роботи найбільш повільного етапу цієї фази.

Як правило, при виконанні реальних програм ядро процесора працює в незбалансованих режимах, коли одна з фаз повільніша в порівнянні з іншою.

Незбалансованість в роботі ядра при виконанні реальних програм обумовлюється як особливостями програм, так і структурними рішеннями ядра процесора і комп'ютера в цілому.

У статті аналіз продуктивності кожної фази моделі ядра виконується при роботі ядра в найбільш сприятливих умовах для структур ядер з наступними параметрами:

1. Коефіцієнт суперскалярності S (у одному такті дешифруються S команд).
2. У ядрі реалізовані сучасні рішення по мінімізації часу перезавантаження конвеєра підготовки команд. Час перезавантаження конвеєра підготовки команд $t_{перезагр} = k\tau$, де:

   $k = 1,2,3,...$ - коефіцієнт, значення якого визначається прийнятими структурними рішеннями по мінімізації часу перезавантаження конвеєра підготовки команд;

   τ - тривалість такту ядра процесора;
3. Розміри буферів між фазами необхідної і достатньої величини.
4. Спеціалізовані функціональні пристрої конвеєрного типу з пропускною спроможністю $1/\tau$.
5. Кеш-пам'ять даних першого рівня здатна читати 2 операнди і записувати один результат кожного такту.

Досліджувані ділянки програм виконуються в найбільш сприятливих з точки зору продуктивності умовах (зокрема, усі дані знаходяться в кеш-пам'яті даних першого рівня).

Основними видами «перешкод» для конвеєра ядра процесора є:

1. Структурні конфлікти.
2. Інформаційні залежності між командами програми.
3. Команди переходів, що викликають перезавантаження конвеєра підготовки команд і, відповідно, простої функціональних пристроїв.

Із структурними конфліктами борються шляхом розділення конфліктних одиниць апаратури і збільшенням їх числа. Наприклад, кеш-пам'ять першого рівня розділена на кеш-пам'ять команд і кеш-пам'ять даних.

Найбільше впливають на продуктивність конвеєрних ядер процесорів команди, що виконують операцію редукції в програмі. При виконанні операції редукції в певному пристрої цей конвеєрний функціональний пристрій працює як комбінаційний з пропускною спроможністю $1/n*\tau$, де τ - тривалість такту, а n - кількість тактів, необхідна для повного виконання операції**.**

У реальному конвеєрі за наявності «перешкод» пропускна спроможність дорівнює $1/t_{\max}$, де $t_{\max}$ - час спрацьовування етапу конвеєра з максимальним часом.

Залежно від кількості команд умовних переходів, від кількості команд обробки, співвідношення часів виконання команд обробки і команд звернення в оперативну пам'ять (і ряду інших параметрів) реальна продуктивність ядра процесора може істотно зменшуватися в порівнянні з теоретичною продуктивністю без

наявності «перешкод».

Дослідження ефективності фази підготовки команд моделі

При виконанні ділянки програми з **m** команд, в якій є одна команда умовного переходу (наприклад це команда умовного переходу на початок циклу) підсистема підготовки команд (фаза 1 моделі) згенерує *A* послідовних груп з *S* команд, де *A* - ціле число, що дорівнює:

$A = m/S$ - якщо m ділиться на *S* без залишку;

$A = m/S + 1$ - якщо m ділиться на *S* із залишком.

Часи генерації A послідовних груп команд за умови, що кожна група команд декодується за час складуть:

- Реальний час генерації

$t_{ген}реал = k\tau + A\tau$,

де $k\tau$ - час перезавантаження конвеєра.

- Ідеальний час генерації

$t_{ген}ід = A\tau$.

Продуктивність підсистеми генерації команд обернено пропорційна до часу генерації однієї порції.

Відношення реальної та ідеальної продуктивностей підсистеми підготовки (генерації) команд складе

$$R = \frac{t_{ген}ід}{t_{ген}реал} = \frac{A\tau}{k\tau + A\tau} = \frac{1}{1 + k/A}.$$

У таблиці 1 приведені значення відношень реальної та ідеальної продуктивностей (*R*) підсистеми підготовки команд залежно від значення параметрів *A* та *k*.

Таблиця 1

Відношення реальної та ідеальної продуктивності *R* залежно від значення параметрів *A* та *k*

	R						
	K=1	**K=2**	**K=3**	**K=4**	**K=5**	**K=6**	**K=7**
A=1	0.50	0.33	0.25	0.20	0.17	0.14	0.125
A=2	0.67	0.50	0.40	0.33	0.29	0.25	0.22
A=3	0.75	0.60	0.50	0.43	0.375	0.33	0.3
A=4	0.80	0.67	0.57	0.50	0.44	0.40	0.36
A=5	0.83	0.71	0.625	0.56	0.50	0.45	0.42
A=6	0.86	0.75	0.67	0.60	0.54	0.50	0.46
A=7	0.875	0.78	0.70	0.64	0.58	0.54	0.50
A=8	0.89	0.80	0.73	0.67	0.62	0.57	0.53

З таблиці 1 видно, що зменшення реальної продуктивності підсистеми підготовки команд може бути значним навіть при роботі в ідеальних умовах. Наприклад, для ядра програми «Множення матриць» при *A*=2 реальна продуктивність у разі, коли всі дані розташовані в кеш-пам'яті даних першого рівня складає від 2/3 (при *k*=1) до 0.25 (при *k*=6) від ідеальної продуктивності.

Дослідження ефективності фази виконання команд моделі

Продуктивність фази 2 на відміну від продуктивності фази 1 значно більше залежить від особливостей виконуваної програми.

Нехай в ядрі виконується скалярний добуток вектору-рядка на вектор-стовпець матриць розмірністю *n*n* елементів програми «Множення матриць». Алгоритм скалярного добутку векторів наступний

$$C_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj}.$$

Основна ділянка програми скалярного множення рядка на стовпець на умовній мові асемблера може мати наступний вигляд:

1. LOAD $a_{ik} \to R1$, спосіб адресації при читанні - автоіндексація.
2. LOAD $b_{kj} \to R2$, спосіб адресації при читанні - автоіндексація.
3. MPY $R1 * R2 \to R3$.
4. ADD $R0 + R3 \to R0$.
5. JMP на початок циклу, якщо його потрібно продовжувати.
6. STORE $R0 \to c_{ij}$, спосіб адресації при записі - автоіндексація.

Якщо розмір матриць n досить великий (наприклад, $n \succ 64$), то для оцінки часу виконання ядра програми можна нехтувати часом виконання команд, які готують основний цикл.

Нехай підсистема підготовки команд кожного такту генерує групу з 4-х команд.

Перша група - це команди:

1. LOAD $a_{ik} \to R1$.
2. LOAD $b_{kj} \to R2$.
3. MPY $R1 * R2 \to R3$.
4. ADD $R0 + R3 \to R0$.

Друга група - одна команда переходу на початок циклу. Оскільки в другій групі команд є команда умовного переходу на початок циклу, то відбувається перезавантаження конвеєра підготовки команд і весь цикл роботи підсистеми підготовки команд повторюється.

Час, що витрачається підсистемою підготовки команд на підготовку ділянки програми з 5 команд при S=4 складає

$t_{ген} = k\tau + 2\tau$.

Час виконання циклу підсистемою виконання повністю визначається специфікою виконання програми, а саме:

- Набором виконуваних операцій.
- Кількістю спеціалізованих функціональних пристроїв.
- Типами спеціалізованих функціональних пристроїв (комбінаційні або конвеєрні пристрої).
- Наявністю інформаційних залежностей між командами.
- Наявністю або відсутністю якої-небудь операції редукції в циклі.
- Часом виконання команд (передусім команд звернення в оперативну пам'ять).
- Алгоритмом виконання команд запису в оперативну пам'ять (наскрізний або відкладений запис) та рядом інших особливостей.

Тому отримати в загальному вигляді оцінку часу виконання циклу ядра програми «Множення матриць» підсистемою виконання команд не представляється можливим.

Розглянемо деякі оцінки часу виконання ядра програми «Множення матриць» розміром $n * n$ підсистемою виконання команд.

У циклі програми визначення чергового результату команда STORE $R0 \to c_{ij}$ записує в оперативну пам'ять результат n-разового повторення циклу.

Команда ADD $R0 + R3 \to R0$ накопичує результати множення, тобто вона виконує операцію редукції.

При розгляді прикладу ядра програми «Множення матриць» були зроблені припущення, що:

1. всі необхідні дані розміщуються в кеш-пам'яті даних першого рівня;
2. кеш-пам'ять даних першого рівня конвеєрний пристрій, здібний кожного такту приймати дві заявки на читання операндів і одну заявку на запис результатів. Затримка у видачі кожного операнда складає 4τ ;
3. функціональні пристрої множення та складання конвеєрні і здатні починати виконання чергової операції кожного такту. Затримка у видачі кожного результату складає 4τ .

Час виконання одного циклу ядра програми підсистемою виконання визначатиметься повним часом виконання команди ADD $R0 + R3 \to R0$. Функціональний пристрій складання при виконанні цієї команди працює як комбінаційний пристрій з часом 4τ (у даному прикладі).

Кеш-пам'ять даних першого рівня, функціональний пристрій множення, хоча вони і конвеєрні і здатні починати виконання чергової операції кожного такту (а команди LOAD дві в одному такті), вимушені працювати в темпі функціонального пристрою складання з часом 4τ.

Час виконання команд ядра в одному циклі в підсистемі виконання команд в самому кращому випадку (після заповнення конвеєра) складе 4τ .

За цей час підсистема виконає 4 команди. Реальна пропускна спроможність підсистеми виконання дорівнюватиме 1 команді за такт. Пікова пропускна спроможність підсистеми виконання дорівнює 4 командам за такт.

Коефіцієнт використання підсистеми виконання команд при виконанні розглянутого прикладу складе 0.25, причому цей результат справедливий для найбільш сприятливого поєднання умов виконання ядра програми і є максимальною оцінкою. Реальний час виконання ядра програми буде більший через те, що:

1. не врахований час виконання підготовчих команд для кожного циклу;
2. не врахований час входу і виходу в конвеєрний режим при обчисленні чергового елементу результату;
3. розмірності матриць такі, що всі необхідні дані не поміщаються в кеш-пам'ять даних першого рівня. А в цьому випадку часи виконання команд звернення в оперативну пам'ять будуть значно більше часу звернення в кеш-пам'ять першого рівня.

Це означає, що при виконанні розглянутої ділянки програми «Множення матриць» (і всієї програми «Множення матриць») реальна пропускна спроможність підсистеми виконання команд не перевершує 1ком/такт, а реальний коефіцієнт використання ядра процесора не перевершує 0.25.

Якщо розглянути сценарій виконання програми «Множення матриць» при таких розмірах матриць, коли необхідно завжди звертатися в оперативну пам'ять за вектором-стовпцем матриці, то час виконання програми практично повністю визначається часом роботи з оперативною пам'яттю. Із-за інформаційної залежності між командами циклу функціональні пристрої «Помножити» і «Скласти» працюють як комбінаційні пристрої з часом кожен, наприклад, 4 такти. В цьому випадку час виконання однієї ітерації з n необхідних для обчислення одного елементу результату складе не менше 270 тактів (час звернення в оперативну пам'ять - не менше 60 нс, частота ядра процесора - 4 Ггц, повні часи множення і складання - 2*4=8 тактів, при читанні рядка першої матриці з оперативної пам'яті читається блок з 64-х байтів, що містить 16 чисел з плаваючою точкою).

В результаті коефіцієнт корисного використання ядра процесора, розрахований як відношення корисного часу зайнятості ядра в одній ітерації до загального часу виконання однієї ітерації складе близько 4.4%.

Дослідження програми рішення диференціальних рівнянь в приватних похідних

Диференціальні рівняння в приватних похідних є широко вживаним математичним апаратом при розробці моделей в самих різних галузях науки і техніки.

Одним з найбільш поширених підходів чисельного рішення диференціальних рівнянь є метод кінцевих різниць *(метод сіток*). Наслідуючи цей підхід, область рішення *D представляється у вигляді дискретного (як правило, рівномірного) набору (сітки) точок (вузлів*).

Використовуючи *п'ятиточковий шаблон для обчислення значень похідних, можна представити рівняння Пуассона в такій формі*

$$\frac{u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}-4u_{ij}}{h^2}=f_{ij}.$$

Це рівняння може бути розв'язане відносно u_{ij}:

$$u_{ij}=0.25(u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}-h^2 f_{ij}).$$

Основна ділянка програми визначення чергової ітерації (ядро цієї програми) на умовній мові асемблера може мати наступний вигляд:

LOAD $U_{i-1,j}^{k-1} \to R1$,
LOAD $U_{i+1,j}^{k-1} \to R2$
ADD $R1+R2 \to R3$
LOAD $U_{i,j-1}^{k-1} \to R4$,
LOAD $U_{i,j+1}^{k-1} \to R5$
ADD $R4+R5 \to R6$
ADD $R3+R6 \to R7$
LOAD $h^2 * f_{i,j-1}^{k-1} \to R8$
SUB $R7-R8 \to R9$
MPY $0,25*R9 \to R10$
LOAD $U_{i,j}^{k-1} \to R11$
SUB $R10-R11 \to R12$
STORE $R10 \to U_{i,j}^{k}$
STORE $R12 \to d_{i,j}$
JMP початок циклу

Всього в наведеному спрощеному прикладі ядра програми визначення чергового наближення значення шуканої функції 15 команд.

Теоретичний час генерації цих 15 команд підсистемою підготовки команд складе

$t_{ген}=k\tau+(15/S)\tau$.

При S=4

$t_{ген}=k\tau+4\tau$.

Умови виконання ядра програми в підсистемі виконання найсприятливіші:

1. Всі масиви знаходяться в кеш-пам'яті даних першого рівня.
2. З кеш-пам'яті даних першого рівня кожного такту можна здійснювати 2 читання і 1 запис.
3. Функціональні пристрої множення і складання конвеєрні з пропускною спроможністю $1/\tau$.

Команда ADD $R1+R2 \to R3$ може починатися тільки після завершення попередніх команд LOAD $U_{i-1,j}^{k-1} \to R1$ **та** LOAD $U_{i+1,j}^{k-1} \to R2$, тобто, не раніше моменту 4τ **.**

Команда ADD $R4+R5 \to R6$ може починатися тільки після завершення попередніх команд LOAD $U_{i-1,j}^{k-1} \to R1$ та LOAD $U_{i+1,j}^{k-1} \to R2$, тобто, не раніше моменту 5τ **.**

Команда ADD $R3+R6 \to R7$ може починатися тільки після завершення попередніх команд ADD $R1+R2 \to R3$ та ADD $R4+R5 \to R6$**,**, тобто, не раніше моменту 10τ **.**

Команда SUB $R7-R8 \to R9$ може починатися тільки після завершення попередніх команд LOAD $h^2 * f_{i,j-1}^{k-1} \to R8$ та ADD $R3+R6 \to R7$ **,** тобто, не раніше моменту 14τ **.**

Команда MPY $0,25*R9 \to R10$ може починатися тільки після завершення попередньої команди SUB $R7-R8 \to R9$, тобто, не раніше моменту 18τ .

Команда SUB $R10-R11 \to R12$ може починатися тільки після завершення попередніх команд MPY $0,25*R9 \to R10$ та LOAD $U_{i,j}^{k-1} \to R11$**,** тобто, не раніше моменту 22τ **.**

Команда STORE $R12 \to d_{i,j}$ може починатися тільки після завершення попередньої команди SUB $R10-R11 \to R12$, тобто, не раніше моменту 26τ .

Команда STORE $R12 \to d_{i,j}$ може завершитися до моменту 26τ **.**

Разом, повний час виконання ядра програми в підсистемі виконання команд складе 26τ . З урахуванням паралельного виконання команд LOAD і STORE з арифметичними командами ADD, MPY, SUB час виконання ядра програми складе в підсистемі виконання команд 20τ **.**

За цей час в ідеальних умовах можуть бути виконані 15 команд. Пропускна спроможність підсистеми виконання команд складе 15/20 команд/такт, тобто 0.725 команд/такт.

Пропускна спроможність підсистеми підготовки команд складе при:

- К=2 - 15 команд/6τ =2.5 команд/такт.
- К=4 - 15 команд/8τ =1.875 команд/такт.

Підсистема виконання в даному прикладі являється повільнішою в порівнянні з підсистемою підготовки команд. Отже, пропускна спроможність ядра процесора визначається пропускною спроможністю підсистеми виконання і не перевищує 0.725 команд/такт.

Отже, і в даному прикладі реальна пропускна спроможність ядра процесора набагато менша теоретичної пропускної спроможності.

Висновки.

1. При виконанні програм існують такі основні причини суттєвої відмінності реальної продуктивності ядра процесора від пікової продуктивності:
   - низька продуктивність оперативної пам'яті в порівнянні з продуктивністю ядра процесора;
   - наявність команд переходу в програмах;
   - наявність інформаційних залежностей між командами програми;
   - наявність команд, які виконують операції редукції.
2. Час роботи підсистеми підготовки команд ядра процесора при генерації однієї ітерації циклу програми визначається як $t_{ген} = k\tau + A\tau$, де:

$k\tau$ - це час перезавантаження конвеєра підсистеми підготовки команд. Значення коефіцієнта k визначається прийнятими структурними рішеннями по мінімізації часу перезавантаження конвеєра підготовки команд.

A - кількість груп команд по *S* в кожній в досліджуваній ділянці програми;

S - ступінь суперскалярності в роботі підсистеми підготовки команд.

3. Час роботи підсистеми виконання команд ядра процесора істотно залежить як від властивостей виконуваних програм (від кількості інформаційно залежних команд, наявності операцій редукції), так і від часу виконання команд, насамперед команд звернення до оперативної пам'яті.
4. При виконанні багатьох реальних програм час роботи підсистеми виконання команд ядра, як правило, значно перевершує час роботи підсистеми підготовки команд, а значить, реальна продуктивність ядра процесора визначається продуктивністю підсистеми виконання команд.
5. Реальна продуктивність підсистеми виконання команд ядра процесора при виконанні ядра програми «Множення матриць» складає не більше 1 команди за такт, а ефективність не перевищує 0.25.
6. Реальна продуктивність ядра процесора може бути меншою теоретичної продуктивності ядра у багато разів.

Література

1. Володимир Воєводін. Суперкомп'ютери і парадокси неефективності. Відкриті системи. СУБД. № 10. 2009.
2. Андрій Созінов. Пікова теоретична продуктивність - це брехня. 03.09.2019.

Постійний URL: http://servernews.ru/993476.
3. Андрій Созінов. Intel Sunny Cove: мікроархітектура процесорів наступного покоління представлена офіційно. 12.12.2018. Постійний URL: https://3dnews.ru/979520.
4. Ілля Гавриченков. Від Sandy Bridge до Coffee Lake: порівнюємо сім поколінь
Intel Core i7. 28.05.2018. Постійний URL: https://3dnews.ru/969891.
5. Ігор Осколков. Знайомство з Intel Xeon Skylake-SP. 11.07.2017. Постійний URL: https://servernews.ru/955164.
6. Ілля Гавриченков. Intel Skylake:Подробиці про мікроархітектуру. 24.08. 2015.
Постійний URL: https://3dnews.ru/919036.
7. Ілля Гавриченков. Огляд процесора AMD Ryzen 7 3700X: Zen 2 у всій красі. 07.07. 2019. Постійний URL: https://3dnews.ru/990334/obzor-amd-ryzen-7-3700x.
8. Ілля Гавриченков. Мікроархітектура Zen 2. 19.06.2019.
Постійний URL: https://3dnews.ru/989344/mikroarhitektura - zen - 2/page - 2.html.
9. Недзельський Д.О. Дослідження ефективності одноядерних суперскалярних обчислювальних систем. Луганськ. Вісник Східноукраїнського національного університету ім. В. Даля. - 2011. - №15(169) Ч. 2. - с. 133-140.
10. Недзельський Д.О. Дослідження і аналіз ефективності структурних методів компенсації впливу команд преходов на продуктивність конвеєрних ядер процесорів. Луганськ: Вісник Східноукраїнського національного університету ім. В. Даля, 2013. - №16(205), Частина 2. - С. 174-181.
11. Недзельський Д.О. Дослідження ефективності підсистеми генерації команд в ядрах сучасних процесорів. Луганськ: Вісник Східноукраїнського національного університету ім. В. Даля, 2017. - №8(238), - С. 64-66.

R e f e r e n c e s

1. Vladimir Voevodin. Supercomputers and the paradoxes of inefficiency. Open systems. DBMS. No. 10. 2009.
2. Andrew Sozinov. Peak theoretical performance is a lie. 03.09.2019.
Permanent URL: http://servernews.ru/993476.
3. Andriy Sozinov. Intel Sunny Cove: the next generation of microarchitecture is officially announced. 12/12/2018. Permanent URL: https://3dnews.ru/979520.
4. Ilia Gavrichenkov. From Sandy Bridge to Coffee Lake: Compare seven generations
Intel Core i7. 28.05.2018. Permanent URL: https://3dnews.ru/969891.
5. Igor Oskolkov. Introducing the Intel Xeon Skylake-SP. 11.07.2017.
Permanent URL: https://servernews.ru/955164.
6. Ilia Gavrichenkov. Intel Skylake: Microarchitecture Details. 24.08. 2015.
Permanent URL: https://3dnews.ru/919036.
7. Ilia Gavrichenkov. AMD Ryzen 7 3700X: Zen 2 CPU review in all its glory. 07.07. 2019. Permanent URL: https://3dnews.ru/990334/obzor-amd-ryzen-7-3700x.
8. Ilia Gavrichenkov. Zen 2. Microarchitecture 19.06.2019.
Permanent URL: https://3dnews.ru/989344/microarchitecture - zen - 2 / page - 2.html.
9. Nedzelskyi D.O. Investigation of the efficiency of single-core superscalar computing systems. Lugansk. Bulletin of the East Ukrainian National University. V. Dalia. - 2011. - №15 (169) Part 2. - p. 133-140.
10. Nedzelskyi D.O. Research and analysis of the effectiveness of structural methods of compensation for the impact of transition teams on the performance of the pipelined cores of processors. Lugansk: Bulletin of the East Ukrainian National University. V. Dahl, 2013. - №16 (205), Part 2. - P. 174-181.
11. Nedzelskyi D.O. Investigation of efficiency of command generation subsystem in kernels of modern processors. Lugansk: Bulletin of the East Ukrainian National University. V. Dahl, 2017. - №8 (238), - pp. 64-66.

Nedzelskyi D., Safonova S. About real productivity and efficiency of nucleus of modern processors

The article discusses the reasons for the mismatch between the peak and real performance of computers, change ranges of real productivity and real efficiency of modern processors cores are investigated, in the presence in such programs of such "obstacles" for the processor kernel pipeline as information dependencies, reduction operations, commands of transitions. A processor core model has been developed. The work imbalance of the core, while executing real programs, is caused by both the features of the programs and the structural decisions of the processor core and the computer altogether. In the article, the performance analysis of each core model phase is performed when the core is operating under the most favorable conditions for core structures. The influence of "obstacles" in the form of transition instructions, information dependencies between the teams, reduction operations on the actual performance and efficiency of the processor core is considered using the examples of the "Matrix Multiplication" and "Solving Partial Differential Equations by Grid Methods" programs. The actual performance and efficiency of the processor core, while executing the "Matrix Multiplication" program under ideal conditions, is no more than 1 command per cycle and the efficiency is less than 0.25. The running time of the CPU core command subsystem depends essentially on the properties of the executable programs (on the number of information-dependent commands, the presence of reduction operations), and on the execution time of the commands, first of all, the memory access commands. While executing many real programs, the processor core performance is determined by the performance of the command execution subsystem. The main reasons

for this are information dependencies between program commands and reduction operations. Actual processor core performance may be many times lower than theoretical performance.
Keywords: *processor, core, instruction pipeline, performance, efficiency, transition, command execution subsystem*.

Недзельский Д.А., Сафонова С.А. О реальной производительности и эффективности ядер современных процессоров

В статье рассмотрены причины несоответствия пиковой и реальной производительности компьютеров. Разработана модель ядра процессора. Рассмотрено влияние «помех» работе конвейера в виде команд переходов, информационных зависимостей между командами, операций редукции на реальную производительность и эффективность ядра процессора на примерах программ «Умножение матриц» и «Решение дифференциальных уравнений в частных производных методом сеток». Реальные производительность и эффективность ядра процессора при выполнении ядра программы «Умножение матриц» в идеальных условиях составляют не более 1 команды за такт и эффективность - менее 0.25. При выполнении многих реальных программ производительность ядра процессора определяется производительностью подсистемы выполнения команд. Основные причины этого - информационные зависимости между командами и операции редукции. Реальная производительность ядра процессора может быть меньше теоретической производительности во много раз.
Ключевые слова: *процессор, ядро, конвеер команд, производительность, эффективность, переход, подсистема выполнения команд*.

Недзельський Д.О. – к.т.н., доцент, доцент кафедри комп'ютерних наук та інженерії Східноукраїнського національного університету імені Володимира Даля, e-mail: nedzelsky946@gmail.com

Сафонова С.О. – к.т.н., доцент, доцент кафедри комп'ютерних наук та інженерії Східноукраїнського національного університету імені Володимира Даля, e-mail: safonovasa@ukr.net.