ПАРАЛЕЛЬНЕ ПРОГРАМУВАННЯ

- Цикл: Професійно-орієнтованих дисциплін
- Спеціальність 6.050103 «Програмна інженерія
- Автор: Качко Олена Григорівна, професор кафедри Програмної інженерії
- elena.kachko@nure.ua

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДИСЦИПЛІНИ

Семестр	Лекції	Практичні заняття	Лабораторні роботи (годин/балів)	Самостійна робота	
				Тест	Інд зав
5	30	10	20 (60)	20 15.11	20 1.12

АРГУМЕНТИ ЩОДО ДИСЦИПЛІНИ

2 приклада.

1 Хай є 4 ядерний процесор. Хай на ньому виконується одночасно 3 програми: 1 – 4 потоки, 2 – 4 потоки, 3 – один потік. Тоді 1 і друга програми отримують по 4/9 процесорного часу, а остання тільки 1/9!! Тому усі сучасні програми треба робити багато поточними, визначати кількість потоків програмно – тільки спеціальними засобами паралельного програмування (**C/C++/Code Generation/Enable Parallel Code Generation/Yes**)!

2 Хай треба заробляти біткоїни (майнити, **«mining»**, - Видобуток корисних копалин) - Знайти такі дані, для яких хеш має визначену форму – немає інших способів, ніж перебір варіантів – успіх в разі паралельного виконання з великою кількістю потоків

МЕТА ВИВЧЕННЯ

полягає у підготовці майбутніх спеціалістів для ефективного використання можливостей сучасних процесорів

Знати:

- рівні паралелізації обчислень для сучасних обчислювальних систем;
- критерії та показники програм в залежності від їх паралелізації;
- типи SIMD команд;
- специфіку розробки паралельних алгоритмів і програм;
- сучасні технології розробки паралельних програм.

Вміти:

- виконувати обчислення показників програм та виконувати їх порівняння;
- визначати тип та характеристики наявного обладнання та обирати найбільш ефективну реалізацію в залежності від визначених характеристик;
- використовувати SIMD команди при програмуванні мовами високого рівня;
- розробляти паралельні алгоритми;
- розробляти паралельні програми за допомогою засобів операційних систем та сучасних технологій;
- виконувати аналіз програм за допомогою сучасних засобів профілювання.

ЛІТЕРАТУРА

Google :

18 400 000 (Parallel programming),

191000 (Параллельное программирование),

125000 (Паралельне програмування)

Основна література

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб: БХВ – Петербург, 2004. – 608 с.
2. Качко Е.Г. Паралельне програмування: Учбовий посібник. – Х.: ХНУРЕ, 2016. – 404 с.
3. Эн. Уильямс. Параллельное программирование на C++ в действии. М. 2012.
4. Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие. – М.: Изд-во МГУ, 2009. – 77 с.

ЛІТЕРАТУРА (ПРОДОВЖЕННЯ)

Навчальні посібники та наукові праці

- Бондаренко М.Ф., Качко Е.Г. Операционные системы. Учебное пособие. Х.: Компания СМІТ, 2006. – 402 с.
- Качко О.Г., Мельникова Р.В. Методичні вказівки до практичних занять з дисципліни „Операційні системи”. Харків, ХНУРЕ, 2009. – 88 с.
- Качко О.Г., Мельникова Р.В. Методичні вказівки до лабораторних занять. Харків: ХНУРЕ, 2009. – 44 с.

Електронні джерела

- Сайти розробників процесорів INTEL (**software.intel.com**, AMD (**developer.amd.com**), IBM (**ibm.com/developerworks**)
- Сайти розробників операційних систем: Microsoft (**http://msdn.microsoft.com/ru-ru/**), Linux (**http://www.linux.com/**), IBM (**http://www.ibm.com/developerworks/**), Hewlett-Packard (**https://developer.palm.com/**), Apple Computer (**http://developer.apple.com/**).

Сайти для освіти

- **http://www.intuit.ru**
- https://stepik.org
- codeacademy.com (русский, английский)
- www.khanacademy.org (английский)
- https://www.coursera.org (английский – з Стэнфорд)
- http://www.programmr.com/ (английский)
- hexlet.org – русский (MAC OC)
- tproger.ru – програмісти для програмістів (найбільш повний огляд курсів для навчання)

РОЗДІЛИ КУРСУ ДЛЯ ВИВЧЕННЯ

Змістовний модуль 1. СУЧАСНІ ПРОЦЕСОРИ ТА ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ

1.1 Архітектура сучасних процесорів

1.2 Критерії та показники програм в залежності від їх паралелізації

1.3 SIMD команди та їх використання при програмуванні мовою високого рівня

Змістовний модуль 2. **РОЗРОБКА ПАРАЛЕЛЬНИХ ПРОГРАМ**

2.1 Розробка паралельних програм

2.2 Паралельні алгоритми

2.3 Технології розробки паралельних програм

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ

Питання для вивчення

- Вступ. Основні визначення
- Класифікація сучасних обчислювальних пристроїв
- Типи паралелізму
- Структура багатоядерного процесору
- Пам’ять і паралелізм

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

Місце	Держава	Ядер	Продуктивність	Тактова частота
1	China, Shanghai	10,649,600	93,014.6	1.45GHz
2	China, Intel (перший в 2016 році)	3,120,000	33,862.7	2.200GHz
3	Switzerland Intel	361,760	19,590.0	2.6GHz

24 роки – другий раз не перші місця USA (1996 - Japan)
3:Shanghai + Intel +Intel ;
Штучний інтелект – перше місце
Large Hadron Collider - Швейцарія
10: Intel 5, IBM(Power) – 2, AMD; China – 1, Japan-1;
500: Intel – 465, Інших - 35

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

Закон Мура: кількість процесорів, і, відповідно продуктивність, подвоюється кожні 24 місяця (якщо врахувати Кількість + підвищення продуктивності – то 18 місяців). В літературі і так і так

Оновлений закон Мура: кількість ядер в чипі подвоюється кожні 24 місяця без збільшення тактової частоти;

3 класа багатоядерних систем:

- **multi-core – Усі ядра на одній мікросхемі, ядро-повнофункціональне, кількість ядер невелика**

(https://en.wikipedia.org/wiki/Microprocessor_chronology#2017s**, Power, IBM. 4 GHz(2017 рік)24 ядер, кожне ядра максимум 8 потоків, всього 24 * 8 = 192 потоків)**

- Multi CPU – окремі обчислювальні блоки – є шини для передачі даних між блоками.
- many-core – для позначення ядер спеціального призначення, кількість ядер сотні і тисячі. Приклад – графічний процесор (2*1536 ядра, усі виконують однаковий набір операцій)

Квантові комп'ютери – особливий клас з супер паралелізмом - **кубіти**

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

- **Паралельні обчислення** – одночасне виконання декількох команд;
- **Паралельна програма** – забезпечує паралельні обчислення.
- **Паралельне програмування –** теорія, прийоми та методи створення паралельних програм.

Проблеми виконання паралельних програм:

- Втрати продуктивності для організації паралельних обчислень – (**Гіпотеза Мінського** –1971 р. прискорення пропорційно двійковому логарифму від кількості процесорів; 32 процесора – прискорення - 5, 1024 - 10) – тут витрати створення потоків, синхронізація,…
- Закон Грошу - (Grosch, 1995 р) – прискорення прямо пропорційно sqrt вартості
- Частина коду в паралельній програмі – послідовно (закони Amdahl, Gustafson);
- Залежність від архітектури;
- Не усі алгоритми можна виконувати паралельно (ітераційні).

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

Теорія та практика паралельного програмування включає в себе наступні напрями:

- **вивчення особливостей сучасних обчислювальних систем з боку паралельних обчислень;**
- вивчення та розробка математичного апарату для теоретичного дослідження програм, еквівалентні перетворення та побудова програм з найкращими властивостями по масштабуванню;
- **аналіз ефективності паралельних обчислень. Необхідно мати зручний практичний інструмент для порівняння алгоритмів і варіантів реалізації по їх ефективності;**
- **вивчення існуючих технологій створення паралельних програм, розробка нових системних програм (операційні системи, компілятори, інтерпретатори, аналізатори, та ін.);**
- **створення паралельних програм для визначених прикладних галузей**

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

МОВИ ПРОГРАМУВАННЯ для створення паралельних програм (Спеціальні мови)

1. **Перша мова паралельного програмування APL (**A Programming Language – 1962 рік**)**
2. **Мова програмування Ada (1980 р.),** для створення програм для керування воєнними об'єктами, надійність та паралелізм. Недоліки. Дуже складний, не інтегрувався з іншими мовами. Сьогодні **Ada – 2012,** проблема інтеграції вирішена
3. **Go (GoLang -** https://golang.org, 2007, Google, перша версія 2009, C подібний), сьогодні підтримується усіма найбільш популярними ОС, в тому числі мобільними, остання версія – травень 2017.
4. **X10 (IBM)** – Java подібний (http://x10-lang.org/)-липень 2004, паралельне програмування для розподілених систем, остання версія 30.06.2017 р.
5. **Milk (MIT – Mac. Техн. Інститут, 2017)** для великих даних – немає принципу локальності – дані, які обробляються, можуть бути далеко друг від друга.

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

МОВИ ПРОГРАМУВАННЯ для створення паралельних програм

Функціональні мови

Усі мови –для паралельного програмування. Чому?

Основна ідея – програма – це функція, яка приймає в якості параметрів теж функції. В якості функцій використовуються виключно чисті функції, тобто немає побічного ефекту →усі параметри можна обчислювати паралельно без усяких перевірок, усі функції можна обчислювати паралельно, якщо їх параметри не залежні.

Приклади мов.

Пролог, Симула, Смолток, F#.

Erlang (Эрла́нг) - спеціально для створення розподілених обчислювальних систем для систем реального часу

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

МОВИ ПРОГРАМУВАННЯ для створення паралельних програм

1. C++, Java, C#, FORTRAN - використання потоків на рівні бібліотек. C++ 11 (C++0x)-2010 р. – підтримка багато поточних програм. Розширення: (for – parallel.for) – сам програміст визначає, який цикл виконувати паралельно. Більшість – для багатопроцесорних систем з розподіленою пам'яттю.
2. **Python** – Для найшвидшого створення коду програмістом та покращення читаємості коду. По продуктивності програм близки до Java. Підтримує різні технології (структурного, об'єктна – орієнтованого, функціонального програмування), тому паралельність закладено в саму мову.

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

Сайти з визначенням рейтингу мов програмування

1. **(https://www.facebook.com/WeblancerNet/) визначає популярність для розробки Web додатків**
2. **Фірма redmonk. http://redmonk.com/sogrady/2017/06/08/language-rankings-6-17/ ,** GitHub
3. **IEEE Spectrum На основі запитів в пошукових системах. http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages**
4. TIOBE Software.

Інформація https://www.tiobe.com/tiobe-index/programming-languages-definition/

На основі пошуків: https://www.tiobe.com/tiobe-index/ (пошук +"<language> programming")

СТАТИСТИКА ПО ВИКОРИСТАННЮ МОВ ПРОГРАМУВАННЯ

	2015	2016	2017
• Java	19.3%	19.01	12.96
• C	14.7%	11.3	6.48
• C++	7.7%	5.8;	5.55
• C#	4.8%	4.91;	4.19
• Python	4.4%	4.4	3.69
• Visual Basic Net	2.13%	2.55	2.6
• PHP	2.7%	3.17.	2.29
• Java script	3.51	2.7	2.098
• Ruby	0.87	1.36	1.96
	60%	52%	40%

Чому????

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

Операційні системи та паралельне програмування

- **UNIX** - спочатку використовувались для Супер-ЕОМ, при розробці її ядра враховувались особливості роботи з багато процесорними системами, і операційна система справляється з задачею ефективного розподілу навантаження для систем, маючих до 256 процесорів;
- **OS Windows** (MS DOS) Сучасна 64-бітна OS Windows 10 підтримує до 64 ядер, але з задачею максимально ефективно розподіляти навантаження між ядрами процесора справляється погано;

(HPC – High Performance Computing)

• Mac OS – головна увага – графічний інтерфейс, для великих обчислювальних систем не використовується.

Linux – (497)- 99%, OS (UNIX) – 3

АРХІТЕКТУРА СУЧАСНИХ ПРОЦЕСОРІВ. ВСТУП. ОСНОВНІ ВИЗНАЧЕННЯ

Розширення компіляторів.

Транслятор сам визначає ділянки коду, які можна виконувати паралельно. VS2013, Properties→C/C++→Code Generation→Enable Parallel Code Generation; Properties→C/C++→Command Line і задати */*Qpar-report:2

Програміст визначає код для паралельного виконання та задає його за допомогою директив компілятора, компілятор формує відповідний код (Open MP)

Додаткові бібліотеки.

Бібліотеки для реалізації паралельного коду (звернення до функцій бібліотеки виконує або сам програміст, або компілятор) . Приклади бібліотек:

PPL, AAL (Библиотека параллельных шаблонов, VS 2013)

TPL (Task Parallel Library для C#).

КЛАСИФІКАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРИСТРОЇВ

Flynn – 1972 г.

Для класифікації використовувались 2 розмірності: команди () та дані.

1. **Single Instraction Single Data** (SISD) – процесори до Pentium.
2. **Single Instraction Multiple Data Data** (SIMD) - процесори з MMX, SSE командами.
3. **Multiple Instraction Single Data** (MISD) – конвеєри.
4. **Multiple Instraction Multiple Data** (MIMD)- багатоядерні процесори.

(Single THREAD Multiple Data – STMD) – для графічних процесорів

Додаткова класифікація (Джонсона):

Multiprocessors – системи з загальною памяттю:

Multicomputers – системи з розподіленою памяттю

	Single Data	Multiple Data
Single Command	SISD	SIMD
Multiple Command	MISD	MIMD

КЛАСИФІКАЦІЯ ОБЧИСЛЮВАЛЬНИХ ПРИСТРОЇВ

Класифікація Джонсона – основана на типах пам'яті, яка використовується, та засобах взаємодії між процесорами

GMSV (global-memory-shared-variables)

(Multiprocessors)

DMMP (distributed-memory-message-passing)

ТИПИ ПАРАЛЕЛІЗМУ

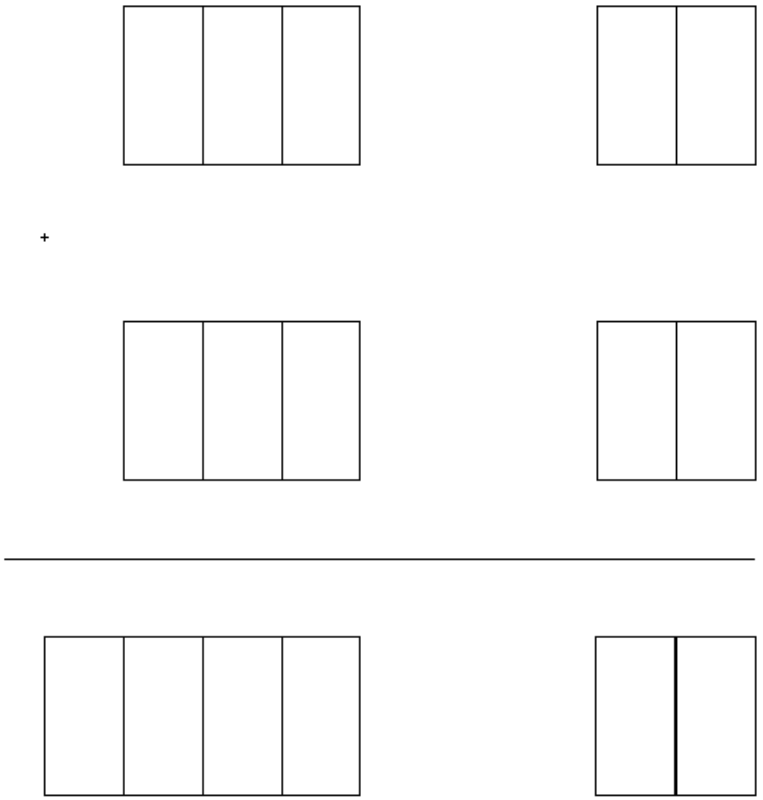
1. Паралелізм на рівні бітів.
2. Паралелізм на рівні команд. Конвеєр.
3. Паралелізм на рівні команд. Суперскалярність.
4. Паралелізм на рівні даних (цикли обробки даних).
5. Паралелізм на рівні функцій (функції. Які виконуються паралельно).
6. Паралелізм на рівні програм

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ БІТІВ

Визначається розрядною сіткою ОС.

Приклад:

Операція складання z=x+y;



ТИПИ ПАРАЛЕЛІЗМУ. КОНВЕЄР

- Перший конвеєр (ATLAS, 1962)
- Конвеєр Pentium (5 стадій)

Приклад:

Mov eax, [x]; 1

Add eax, [y]; 2

Add eax, [z]; 3

Add eax, [u]; 4

Mov [v], eax; 5

Start Time	Ком	Код	Оп1 Оп2	Операц	Зап	End Time
1	1	Mov	Eax x	Mov	Eax	5
2	2	Add	Eax y	Add	Eax	6
3	3	Add	Eax z	Add	Eax	7
4	4	Add	Eax u	Add	Eax	8
5	5	Mov	Eax v	Mov	v	9

ТИПИ ПАРАЛЕЛІЗМУ. КОНВЕЄР

Хай конвеєр має *m* пристроїв, час виконання однієї команди 1, визначити час виконання *n* команд без конвеєра та з конвеєром (витратами, пов'язаними з використанням конвеєра, зневажити)

- Без конвеєра $T_s = n$.
- З конвеєром: $T_p = 1 + (n-1)/m$.
- Якщо m = 5, n=100, отримаємо: T_s = 100; T_p =20.8

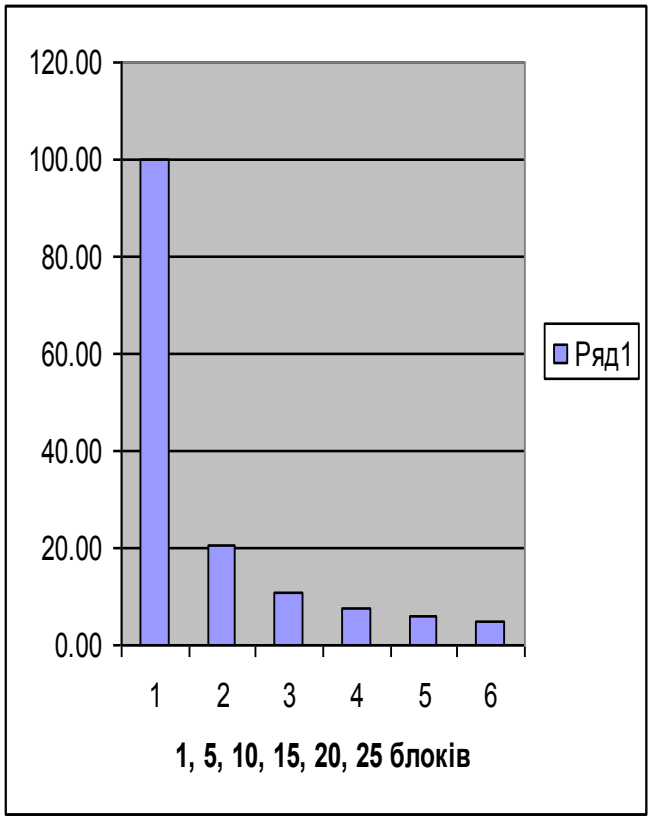
Чим більше блоків m, тим більше виграш!!!

Pentium 4 конвеєр має 25 блоків (останнє значення на діаграмі)

Особливості команд серіалізації (cpuid)- перед виконанням закінчується виконання усіх попередніх команд

Рекомендація. Перед виміром часу виконання використовувати команду серіалізації!!!

Конвеєр та команди переходів.



ТИПИ ПАРАЛЕЛІЗМУ. КОНВЕЄР

Конвеєр та команди переходів. Хай конвеєр має 5 блоків.
Конвеєр та команди переходів. Хай конвеєр має 5 блоків.
c = max (a, b)
moveax, [a]
cmp eax, [b]
jge short m1
moveax, [b]
mov[c]. eax

mov eax, [a]				
cmp eax, [b]	mov			
jge short m1	Cmp	eax, [a]		
Mov eax, [b]	Jge	eax, [b]	Eax , [a]	
Mov [c]. Eax	*Mov*	*M1*	Eax><== [b]	Eax = [a]
	Mov	*eax, [b]*	M1	eflags
		[c]. eax	*Eax , [b]*	**Goto!!!!**
			[c]. , eax	Eax = [b]
				[c]. , eax
Розмір конвеєру:більше 20 для останніх одноядерних (Pentium 4), 14 для i7				

ТИПИ ПАРАЛЕЛІЗМУ. КОНВЕЄР

БПП. Статичне передбачення

- Виконується для команди переходу, яка виконується вперше.
- Результат заноситься в БПП.
- Передбачається відсутність переходу для переходу вперед і наявність для переходу назад

БПП. Динамічне передбачення

- Виконується для команд переходу, інформація для якої в БПП.
- Інформація коректується.
- Перехід прогнозується в залежності від попередніх результатів

Блок передбачення:

Адреса команди переходу, флаги (адреса – тільки молодші біти)

ТИПИ ПЕРЕХОДІВ

безумовні

- 1 goto (jmp label)
- Switch (jmp label[reg])
- Віртуальні функції

```
Mov    eax, tabl[reg]
Jmp    eax
```

Умовні

if(a) block

```
if (!a) goto lab1;
block
lab1:
```

if(a)block1; else block26

```
if(!a)goto lab1;
block1; goto lab2;
lab1:
block2
lab2:
```

ТИПИ ПЕРЕХОДІВ

Цикли

```
for (e1;e2;e3) block;
e1;
goto lab1;
lab2:
e3;
lab1:
If (!e2) goto lab3;
block;
goto lab2;
Lab3:
```

```
while (e1)block;
if (!e1)goto lab1;
block;
lab1:
```

```
do {block }while (e1);
lab1:
block;
if (e1) goto lab1;
```

АЛГОРИТМИ ПЕРЕДБАЧЕНЬ ДЛЯ УМОВНИХ ПЕРЕХОДІВ

Pentium

2 біта.

Передбачається відсутність переходу, якщо обидва біта -0, наявність. Якщо хоч би один біт – 1

Біт встановлюється в 1, якщо перехід є.

Біт встановлюється в 0, якщо переходу немає.

Будемо позначати 1 – якщо перехід є, 0 – якщо немає.

E (error) – помилка в передбаченні, O (ok) – правильне передбачення.

Приклад 1

11011101. Якщо немає передбачень – 6 помилок

EOEOOOEO. Є передбачення – 3 помилки

Найгірший варіант: 10101010101

АЛГОРИТМИ ПЕРЕДБАЧЕНЬ ДЛЯ УМОВНИХ ПЕРЕХОДІВ

Pentium 2

4 біта для передбачення. В 1 встановлюється не довільний біт, а відповідний.

```
Int cur = 0;
Int count [4];
for (i = 0, j = 0; i < n; i++){
        j++;

        if (j == 4)j = 0;
        if (cif [i] ) count[j] = 1; else count[j] = 0;
}
```

Приклад

11011101	10101010101010
eeoeoooo	eoeooooooooooo

Добре працює, якщо є період 4

Але якщо є період, можна команди переходу не використовувати!!!

УНИКНЕННЯ ПЕРЕХОДІВ В РАЗІ НАЯВНОСТІ ПЕРІОДУ

хай необхідно запрограмувати формулу:

y[i]=f1(x[i]) i = 0, 3, 6,…3k

y[i]=f2(x[i]) i = 1, 4, 7,…3k+1

y[i]=f3(x[i]) i = 2, 5, 8,…3k+2

1 варіант

```
for (i = 0, k = 0; i < n; ++i, k++){
        if (i == 3 * k) y [i] = f1 (x[i]); else
        if (i == 3 * k + 1) y [i] = f2 (x[i]); else y [i] = f2 (x[i]);
```

}період не кратний 4 – передбачення погане. Є else, тому помилки кожного разу.

2 варіант

```
for (i = 0; i < n; i +=3){
        y [i] = f1 (x[i]); y [i + 1] = f2 (x[i + 1]); y [i + 2] = f3 (x[i+2]);
}
if (i < n) {y [i] = f1 (x[i]); i++;
if (i < n) {y [i] = f2 (x[i]);
```

В подальшому циклічні переходу не розглядаються!!!

АЛГОРИТМИ ПЕРЕДБАЧЕНЬ ДЛЯ УМОВНИХ ПЕРЕХОДІВ

- **Алгоритм з рахівником з насиченням**

Розглянемо на прикладі рахівника з 4 бітами:

```
BYTE bFlags = 1;
// Прогнозування
BOOL Prediction (BYTE bFlags){
        return bFlags >=2;
} // Встановлення бітів
BYTE SetFlags (BYTE bJMP, BYTE bFlags){
 if (bJMP){
        if (bFlags <8)
                bFlags <<=1;
}else
         if (bFlags >1)
                bFlags >>=1;
}
return bFlags;}
```

```
100001
1 0001 E 0010
0 0010 O 0001
0 0001 O 0001
0 0001 O 0001
0 0001 O 0001
1 0001 E 0010          2/3
-------------------------
11010101
1       0001    E       0010
1       0010    E       0100
0       0100    O       0010
1       0010    O       0100
0       0100    E       0010
1       0010    E       0100
0       0100    E       0010
1       0010    E       0100
¼ - OK
```

АЛГОРИТМИ ПЕРЕДБАЧЕНЬ

- 2 – рівневе адаптивне передбачення

Таблиця станів, кількість елементів визначається n (n = 2 – 4 стани).

Для кожного стану ставиться в відповідність регістр з насиченням. Для визначеного стану (попередні n бітів) прогнозується стан згідно його регістру. Стан регістру корегується, якщо потрібно.

Для реальних процесорів n = 4, 16 рядків таблиця, разхівник 2 біта, поточний стан 4 біта, всього 16 * 2 + 4 бітів для кожної команди переходу

(локальний режим).

Глобальний – загальна таблиця

1 0 1 0 1 1 1 0

E O E O O O O O

Поточний стан

0 1

1 0

0 1

1 0

0 1

1 1

1 1

1 0

Таблиця:

00	01
01	01
10	10
11	01

АЛГОРИТМИ ПЕРЕДБАЧЕНЬ

Передбачення для циклів з відомою кількістю виконання.

Окремо для циклів (n < 64 – 6 бітів)

- Передбачення для косвених переходів – одна адреса переходу
- Передбачення для return

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ КОМАНД. СУПЕРСКАЛЯРНІСТЬ

Паралелізм на рівні команд. Суперскалярність.
Що таке суперскалярний процесор?
Є декілька блоків для виконання операцій або навіть декілька конвеєрів

Приклади:
CDC 6600 (1964) - 10 незалежних функціональних пристроїв (конвеєра немає).
CDC 7600 (1969) – 8 незалежних конвеєра.
ILLIAC IV (1974) - план – 256 процесорних елементів, зробили – 64 (дуже дорого).
CRAY-1 (1976) – 12 незалежних конвеєра.
Intel (1993) (не кожна команда потребує повний конвеєр, наприклад, inc eax, 2 блока)

Залежні та незалежні команди

```
X = a;          X= a;                    X= a;          y = c;
z = x + b;      z = x + b; y = c;        z = x + b;
y = c;          u = z + y;               u = z + y;
u = z + y;
```

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ КОМАНД. СУПЕРСКАЛЯРНІСТЬ

Приклад використання суперскалярності

- **Хай необхідно обчислити значення поліному:**

$Y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

Метод Горнера

$Y = (\ldots((a_n x + a_{n-1}) x + a_{n-2})x \ldots + a_1)x + a_0$

При n = 3 маємо:

$Y = a_3 x^3 + a_2 x^2 + a_1 x + a0$ и $Y = ((a_3 x + a_2) x + a_1)x + a_0$.

По методу Горнера необхідно послідовно виконати 3 операції множення та 3 операції додавання незалежно від кількості пристроїв для виконання цих операцій, всього 6 операцій.

Схема обчислення для процесора з 3 пристроями для цілих чисел:

$a_3 x$	$a_2 x$	$a_1 x$
$a_3 x^2$	$a_2 x^2$	$a_1 x + a_0$
$a_3 x^3$	$a_2 x^2 + a_1 x + a_0$	
$a_3 x^3 + a_2 x^2 + a_1 x + a_0$	Всього 4 операції	

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ ДАНИХ. SIMD КОМАНДИ

Паралелізм на рівні даних. SIMD команди

Класи команд:

- **MMX** (64 біт, 2, 4, 8 елементів, цілі, регістри з плаваючою точкою).
- **3DNow!** (64 біт, 2 float, регістры з плаваючою точкою, горізонтальні операції)
- **SSE** (128 біт, int, double, float, 2-16 елементів, розширений набір операцій)
- **AVX** (256 (512, 1024))

ТИПИ ПАРАЛЕЛІЗМУ. ПАРАЛЕЛІЗМ НА РІВНІ ЗАДАЧ

Паралелізм на рівні задач

Потоки **(**визначення**)**

Способи завдання паралелізму на рівні задач

- **Псевдо** паралелізм **(**операційна система, витіснення**)**
- **HYPER Threading (**операційна система, витіснення, якщо кількість потоків більше ніж 2**)**
- **multi-core (**операційна система, витіснення, якщо кількість потоків більше кількості ядер**)**
- **many-core** (операційна система, спеціальні бібліотеки, наприклад, CUDA технологія)
- **Системи з розподіленою пам'яттю** (Системи з масовим паралелізмом або кластери) – розподілені операційні системи

СТРУКТУРА БАГАТОЯДЕРНОГО ПРОЦЕСОРУ

<table>
<tr><td>Процесорне ядро 1</td><td>Процесорне ядро 2</td><td>Процесорне ядро 3</td><td>Процесорне ядро 4</td></tr>
<tr><td>Кеш L2 (512 КБайт)</td><td>Кеш L2 (512 КБайт)</td><td>Кеш L2 (512 КБайт)</td><td>Кеш L2 (512 КБайт)</td></tr>
<tr><td colspan="4">Кеш L3 (2 / 4 Мбайт)</td></tr>
<tr><td colspan="4">Інтерфейс системних запитів</td></tr>
<tr><td colspan="4">Перехресний комутатор</td></tr>
</table>

ПАМ’ЯТЬ І ПАРАЛЕЛІЗМ

Масивні-паралельні комп'ютери. Система складається з однорідних обчислювальних вузлів, що включають один або кілька центральних процесорів, локальну пам'ять, комунікаційний процесор або мережний адаптер

Симетричні багатопроцесорні системи. Однакові процесори підключаються до загальної пам'яті, при цьому швидкість звертання до загальної пам'яті для всіх процесорів однакова.

Системи з неоднорідним доступом до пам'яті. Система складається з однорідних базових модулів (плат), що складаються з невеликої кількості процесорів і блоків пам'яті. Підтримується єдиний адресний простір, апаратно підтримується доступ до пам'яті інших модулів («чужої» пам'яті). При цьому доступ до локальної пам'яті в кілька разів швидше, ніж до «чужої», а швидкість доступу до «чужої» пам'яті тим менше, чим далі ця пам'ять від процесору, який виконує доступ.

Паралельні векторні системи. Основною ознакою є наявність спеціальних *векторно-конвеєрних процесорів*, у яких передбачені команди однотипної обробки векторів незалежних даних, що ефективно виконуються на конвеєрних функціональних пристроях.

Кластерні системи. Складаються з неоднорідних систем. Пам’ять розподілена

ВИСНОВКИ

- Чому сьогодні треба вчитися розробляти програми з паралельними обчисленнями? – апаратні засоби працюють ефективно тільки в разі таких обчислень!
- Чи задовольняють сучасні ОС вимогам паралельного програмування? – не зовсім, плохо вміють розподіляти навантаження, валикі накладні витрати, .повязані з використанням потоків?
- Чи достатньо перекомпілювати програму для отримання паралельних обчислень? Ні. Сучасні компілятори вміють забезпечити паралельне виконання тільки циклів, і то в разі, якщо гарантується незалежність ітерацій циклу.
- Чи довільний алгоритм можна виконувати паралельно. Ні, навіть найпростіші алгоритми створені з урахуванням послідовного обчислення – треба вчитися розробляти спеціальні алгоритми.
- Чи є мова програмування для створення паралельних програм? – Такі мови є (функціонального програмування), але вони не ефективні для систем з загальною пам'яттю. Сьогодні створюються нові мови. Приклад – мова Axum
- Чи є технології для розробки паралельних програм. Так, і їх вивчення є задачею цієї дисципліни

ПИТАННЯ ДЛЯ САМОСТІЙНОГО ВИВЧЕННЯ

- Багатопроцесорні обчислювальні і операційні системи.
- Обчислювальні системи з загальною та розподіленою пам'яттю.

Паралельне програмування (учбовий посібник)

МАТЕРІАЛИ ДЛЯ ЕКСПРЕС-КОНТРОЛЮ

- Назвіть рівні паралелізму
- До якого рівня паралелізму відноситься конвеєрна обробка команд?
- Виведіть формулу залежності часу виконання *n* команд для конвеєра з *m* вузлами, якщо час виконання однієї команди у випадку відсутності конвеєра дорівнює одиниці. Накладними витратами, пов'язаними з використанням конвеєра, можна зневажити. Зробіть висновки про залежності цього часу від кількості вузлів конвеєра.
- Що таке статичне й динамічне прогнозування переходів, як їх варто враховувати при складанні програм?
- Проаналізуйте код, що формується транслятором для операторів for, while, do, switch і сформулюйте рекомендації з їхнього використання.
- Складіть функцію обчислення найбільшого загального дільника й мінімізуйте кількість команд переходу для цієї функції.
- Напишіть код для обнуління молодших *n* і старших *m* біт в 32 бітному числі, не використовуючи переходів (*m, n* - константи, $m + n < 32$)
- Складіть функцію упорядкування даних методом простої вставки, з мінімальною кількістю команд переходів.
- Розгляньте класичний алгоритм обчислення суми й змініть його таким чином, щоб кількість необхідних тактів була мінімальною для заданої кількості ядер процесора.
- Чим відрізняються системи з розподіленою й загальною пам'яттю, до якого типу систем відносяться системи на основі багатоядерного процесору