МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2019 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження ефективності ядер процесорів з спеціалізованими функціональними пристроями

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 – "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	Д.О. Недзельський (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О. Критська (ініціали, прізвище)
Студент:	_____ (підпис)	В.І. Коннов (ініціали, прізвище)
Група:		КІ-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки "Комп'ютерна інженерія"
(шифр і назва)
Спеціальність 123 – "Комп'ютерна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____2019 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Коннову Віктору Іллічу
(прізвище, ім'я,по батькові)

1. Тема роботи Дослідження ефективності ядер процесорів з спеціалізованими функціональними пристроями
керівник проекту (роботи) к.т.н., доц. Недзельський Д.О.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " __ " __ 2018 року №
2. Строк подання студентом проекту (роботи) 12.01.2019
3. Вихідні дані до проекту (роботи) матеріали науково-дослідницької практики

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд структур ядер сучасних процесорів. 2. Дослідження структури ядра з універсальнимфункціональним пристроєм. 3. Дослідження структури ядра з m спеціалізованими функціональними пристроями. 4 Дослідження впливу перешкод від інформаційних залежностей та команд переходів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)
Презентація доповіді.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Критська Яна Олександрівна		

7. Дата видачі завдання 18.10.2018

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Аналіз стану питання в літературі. Вивчення структур ядер сучасних процесорів.	18.10.2018 – 16.11.2018	
2	Вивчення методів моделювання та вибір методу.	17.11.2018 – 22.11.2018	
3	Розробка моделі ядра з універсальним функціональним пристроєм та її дослідження.	23.11.2018 – 28.11.2018	
4	Розробка моделі ядра з m спеціалізованими функціональними пристроями та її дослідження.	29.11.2018 – 07.12.2018	
5	Дослідження впливу інформаційних залежностей на продуктивність компютера.	08.12.2018 – 12.12.2018	
6	Розробка моделі ядра при наявності перешкод в вигляді команд переходів та її дослідження.	13.12.2018 – 20.12.2018	
7	Розробка заходів з охорони праці.	21.12.2018 – 24.12.2018	
8	Оформлення пояснювальної записки і графічного матеріалу.	25.12.2018 – 06.01.2019	
9	Підготовка та подання магістерської роботи до захисту	07.01.2019 – 12.01.2019	

Студент _____ Коннов В.І.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Недзельський Д.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Коннов В.І. Дослідження ефективності ядер процесорів з спеціалізованими функціональними пристроями

Розглянуті та проаналізовані структури ядер сучасних процесорів. Розроблені моделі ядер: з універсальним функціональним пристроєм; з m спеціалізованими функціональними пристроями; досліджено вплив перешкод на продуктивність комп'ютерів. Отримані аналітичні вирази для коефіцієнтів використання пристроїв. Проведено дослідження ефективності розроблених моделей структур ядра.

Практичне значення, галузь застосування роботи: Результати дослідження можуть бути корисні при оптимальному виборі структури ядра з урахуванням співвідношення як показників ефективності, так і апаратних витрат на реалізацію.

Ключові слова: ЯДРО, ПРОЦЕСОР, КОНВЕЄР, ПЕРЕШКОДИ, КОЕФІЦІЄНТ ВИКОРИСТАННЯ.

АННОТАЦИЯ

Коннов В.И. Исследование эффективности ядер процессоров с специализированными функциональными устройствами

Рассмотрены и проанализированы структуры ядер современных процессоров. Разработанны модели ядер: с универсальным функциональным устройством; с m специализированными функциональными устройствами; исследовано влияние помех на производительность компьютеров. Полученные аналитические выражения для коэффициентов использования устройств. Проведено исследование эффективности разработанных моделей структур ядра.

Практическое значение, область применения работы: Результаты исследования могут быть полезны при оптимальном выборе структуры ядра с учетом соотношения как показателей эффективности, так и аппаратных затрат на реализацию.

Ключевые слова: ЯДРО, ПРОЦЕССОР, КОНВЕЙЕР, ПОМЕХИ, КОЭФФИЦИЕНТ ИСПОЛЬЗОВАНИЯ.

ABSTRACT

Konnov V.I. Investigation of the effectiveness of processor cores with specialized functional devices

The structure of the nuclei of modern processors is considered and analyzed. Designed kernel models: with a universal functional device; with m specialized functional devices; the impact of interference on the performance of computers. The obtained analytical expressions for the coefficients of device use. The research of the efficiency of developed models of structures of the nucleus was conducted.

Practical significance, scope of application: Research results can be helpful in optimally choosing the core structure, taking into account the ratio of both performance indicators and hardware implementation costs.

Keywords: CORE, PROCESSOR, PIPELINE, INTERFERENCE, UTILIZATION FACTOR.

ЗМІСТ

ВСТУП

Стрімкий розвиток науки і проникнення людської думки в усі нові області разом з рішенням поставлених раніше проблем постійно породжує потік питань і ставить нові, як правило складніші, завдання. За часів перших комп'ютерів здавалося, що збільшення їх швидкодії в 100 разів дозволить вирішити більшість проблем, проте продуктивність сучасних супер ЕОМ сьогодні є явно недостатньою для багатьох учених. Гідродинаміка, сейсморозвідка і прогноз погоди, моделювання хімічних сполук, дослідження віртуальної реальності - ось далеко не повний список областей науки, дослідники яких використовують кожну можливість прискорити виконання своїх програм.

Нарощування об'єму кеш-пам'яті і числа команд, що запускаються на виконання, не дають приросту продуктивності процесорів, пропорційного апаратним ресурсам, що витрачаються, і енергії. Сьогодні потрібно абсолютно інші структурні підходи.

Процесор, що виконує кожну команду за чотири такти (вибір команди, декодування і вибір операндів, виконання, запис результату), в принципі, забезпечує корисну роботу тільки в такті "виконання". Для збільшення такої низької (25%) ефективності використовується конвеєрна організація: такти послідовно обраних команд поєднуються в часі так, що в кожен період одна з команд обов'язково знаходиться на етапі "виконання". Для організації конвеєра потрібна додаткова апаратура, але виграш в продуктивності це виправдовує. Для виконання перетворень, що задаються командами процесора, служить сукупність функціональних пристроїв (ФП), таких як цілочисельне арифметико-логічне пристрій (АЛУ), АЛУ з плаваючою точкою і інші. Якщо тривалість перетворення у ФП перевищує один такт, з'являється можливість здійснення іншим ФП (чи тим же ФП - при конвеєрній реалізації) перетворення, що задається наступною командою. При цьому не треба чекати завершення вже виконуваних перетворень, а в результаті обчислення прискорюються.

Процесори, що реалізовують цю можливість, називають "суперскалярними". У них кожного такту декодуються декілька команд. Кожна команда може бути виконана в своєму ФП.

Зменшення тривалості такту, розробка процесорів з багатотактними конвеєрами (у тому числі з сукупністю багатотактних ФП) потенційно забезпечують швидкодію, пропорційну сумарній продуктивності використовуваних ФП. Проте простої пристроїв через неможливість завантажити при виконанні програми обумовлюють зниження продуктивності. Ці простої пов'язані, принаймні, з трьома чинниками: очікування завершення обміну з істотно повільнішою пам'яттю, виконання команд умовних

переходів, залежності між командами - за даними і використовуваними ресурсами (наприклад, регістрам). Для боротьби з простоями, пов'язаними з повільною пам'яттю в обчислювальних системах використовується багаторівнева кеш-пам'ять, використовується розшарування пам'яті. Усе це дозволяє при забезпеченні тимчасової і просторової локалізації звернень до пам'яті створювати ефект швидкого доступу.

Для зменшення втрат продуктивності із-за простоїв ФП, обумовлених визначенням напрямів переходів у виконуваній програмі, служать схеми передбачення переходів і переупорядковування команд (виконання в іншому порядку). Останній метод дозволяє ослабити вплив на час простоїв ФП істинних залежностей між командами читання після запису (обчислення повинні зупинитися на команді, яка використовуватиме ще не отриманий результат іншої команди, - поступила на виконання, але не завершеною).

Сукупність цих і ряду інших прийомів забезпечила підвищення продуктивності сучасних комп’ютерів.

Актуальність теми. За часів перших комп'ютерів здавалося, що збільшення їх швидкодії в 100 разів дозволить вирішити більшість проблем, проте продуктивність сучасних комп’ютерів сьогодні є явно недостатньою для багатьох учених. Тому, дослідження існуючих структур і пошук засобів для підвищення їх ефективності є актуальним.

Об'єктом дослідження магістерської роботи є конвеєрне суперскалярне ядро процесора.

Предметом дослідження – аналітичні, чисельні, імітаційні моделі, необхідні для проведення досліджень комп’ютерних систем.

Метою є дослідження впливу структурних рішень, за допомогою яких забезпечується паралелізм при виконанні команд, на ефективність функціонування обчислювальної системи.

Для досягнення поставленої мети, в магістерській роботі сформульовані та вирішені такі **задачі**:

- проаналізовані структури ядер сучасних процесорів на прикладі процесора фірми Intel зі структурою Haswell та методи дослідження складних структур;
- розроблені аналітичні моделі та імітаційні моделі конвеєрних суперскалярних ядер процесорів. В моделях врахована значна кількість спеціалізованих

функціональних пристроїв для забезпечення апаратного паралелізму при виконанні команд, а також „перешкоди” роботі конвеєра генерації команд.

- приведені результати досліджень.

Методи рішення поставлених задач базуються на використанні ідей систем масового обслуговування, сучасних технологій для створення аналітичних та імітаційних моделей.

Наукова новизна одержаних результатів. Одержали подальший розвиток моделей для досліджень ядер процесорів з універсальними та спеціалізованими функціональними пристроями.

Розроблені моделі для досліджень ядер процесорів з будь-якою кількістю функціональних пристроїв, а також моделі, які враховують вплив «перешкод» роботі конвеєра ядра процесора без блока передбачення переходів. Отримано ряд аналітичних результатів для оцінки ефективності паралелізму в ядрах сучасних процесорів.

Практичне значення. Розроблені моделі дозволяють проводити дослідження в галузі комп’ютерних технологій, реально та обґрунтовано оцінювати доцільність нововведень, які пропонуються при розробці ядер процесорів.

Структура та обсяг магістерської роботи. Магістерська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Робота викладена на 85 сторінках машинописного тексту, містить 75 сторінок основного тексту, 9 рисунків, 19 таблиць, 3 додатоки на 36 сторінках. Бібліографічний список включає 27 найменувань.

РОЗДІЛ 1
АНАЛІЗ СТРУКТУР ЯДЕР СУЧАСНИХ ПРОЦЕСОРІВ ТА МЕТОДІВ ДОСЛІДЖЕННЯ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

1.1 Огляд структур ядер сучасних процессорів

1.1.1 Загальні принципи обробки команд в сучасних процесорах

На рисунку 1.1 представлена структурна схема та організація обробки команд в ядрі сучасного конвеєрного суперскалярного комп’ютера.

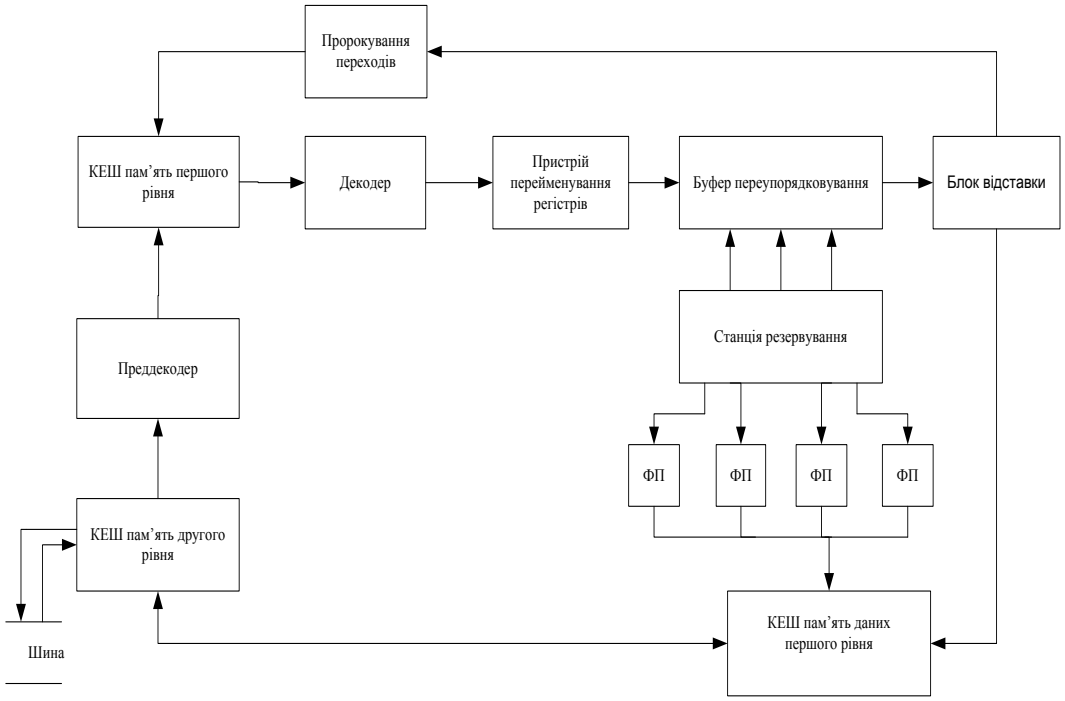


Рис. 1.1 - Загальна організація обробки команд в ядрі сучасного процесора

Частина команд програми, яка "найактивніше" виконується в деякий відрізок часу, розміщена в кеші інструкцій. Залежно від організації процесора, інструкції в цьому кеші можуть зберігатися в початковому незміненому виді, або в частково "декодованому" виді, або в повністю "декодованому" виді (тобто у вигляді готових мікрооперацій). У разі відсутності в цьому кеші потрібних інструкцій (початкових або перетворених) вони прочитуються з кеша другого рівня, при необхідності піддававшись попередньому декодуванню перед поміщенням в кеш інструкцій.

Інструкції прочитуються з кеша інструкцій блоками, з випереджаючою передвибіркою. Поточний блок інструкцій вирушає відразу в два пристрої - декодер інструкцій і передбачувач адресів переходів. Декодер перетворить початкові (або частково декодовані) інструкції в мікрооперації, а передбачувач адресів переходів визначає, чи є в оброблюваному блоці інструкції переходу, чи будуть ці переходи здійснені і по яких адресах. Якщо який-небудь перехід передбачається як "виконуваний", то негайно прочитується блок інструкцій, що знаходиться за передбаченою адресою.

Тим часом знову породжена декодером група мікрооперацій поступає в пристрій перейменування регістрів і виділення ресурсів (Rename/Allocate). Перейменування (чи перепризначення) регістрів - це виділення цій мікрооперації нового екземпляра внутрішнього регістра процесора, куди будуть поміщені результати виконання цієї мікрооперації. Всі подальші операції, які залежать від результатів даної мікрооперації, використовуватимуть цей регістр як операнд. "Перейменований" регістр ставиться у відповідність регістру, який вказаний в машинній інструкції (так званому "архітектурному регістру"). Необхідність в перепризначенні регістрів пов'язана з тим, що архітектурних регістрів зазвичай дуже мало, і при використанні обмеженого числа регістрів неможливо ефективно виконати потік операцій - кожна нова операція, яка використовує певний регістр, була б вимушена чекати завершення всіх попередніх операцій, які до нього звертаються (навіть, якщо їй не потрібні результати цих операцій). Наявність начної кількості машинних (фізичних) регістрів і механізму перейменування дозволяє обійти цю проблему. Інформація про відповідність регістрів зберігається в спеціальних таблицях. У момент відставки мікрооперації буде проведено зворотне перетворення фізичного регістра в архітектурний.

Після перетворення та підготовки регістрів група мікрооперацій, що поступила, записується в кінець спеціальної черги, яка називається "буфер переупорядкування" (ReOrder Buffer, ROB). Ця структура є ключовою в організації позачергового виконання операцій. В ній зберігаються всі мікрооперації і необхідні допоміжні дані від моменту завершення декодування і виділення ресурсів до моменту відставки. Таким чином, довжина буфера переупорядкування обмежує число операцій, які одночасно можуть знаходитися в виконавчій частині ядра процесора (підсистемі позачергового виконання), - від "найстарішої", яка ще не завершена і тому не може "піти у відставку", до "найновішої", яка тільки що поступила з декодера. У разі переповнення буфера переупорядкування, робота декодера припиняється доти, поки не відбудеться відставка операцій на початку черги і звільнення місця для нових мікрооперацій.

Одночасно з попаданням в буфер переупорядкування нова група мікрооперацій передається в іншу структуру даних, звідки мікрооперації відсилатимуться на виконання безпосередньо у функціональні пристрої. Ця структура, відома під назвою "пункт резервування" або "резервація" (Reservation Station, RS), є один або декілька буферів, до яких приєднані ці функціональні пристрої. У кожному такті в цих буферах здійснюється пошук операцій, які готові до виконання (тобто, аргументи яких вже вичислені або обчислюються і будуть готові до моменту попадання операції у функціональний пристрій) незалежно від порядку, в якому вони записувалися в буфери. Пристрій, який здійснює цей пошук і запуск на виконання, називають планувальником, а самі буфери - чергами планувальника. Планувальник відстежує залежності між операціями за даними і прогнозує готовність операцій до виконання в пристроях.

Таким чином, пошук мікрооперацій для позачергового виконання завжди здійснюється тільки в межах такого буфера планувальника (єдиного для всіх функціональних пристроїв, або специфічного для кожної групи пристроїв), і цей буфер виглядає як "вікно", в якому (при необхідності) відбувається зміна порядку виконання операцій.

Черга планувальника є повністю асоціативним буфером, з точки зору пошуку операцій для виконання. Але з точки зору запису нових мікрооперацій, вона поводиться як звичайна черга. Всі мікрооперації, що знаходяться в якийсь момент в чергах планувальника, одночасно знаходяться і в буфері переупорядкування. При запуску операції на виконання у функціональному пристрої відповідна мікрооперація, що знаходиться в черзі планувальника, видаляється з нього, а у момент завершення операції робиться позначка у відповідному елементі буфера переупорядкування. Коли усі мікрооперації, які передують данній, успішно виконаються і відправляться у відставку, ця мікрооперація також може бути відставлена і видалена з буфера переупорядковування. Проте може виявитися, що мікрооперація потрапила в помилкову (спекулятивну) гілку виконання із-за невірного передбачення переходу - в цьому випадку вся гілка буде видалена з буфера переупорядкування після правильного виконання і відставки цієї інструкції переходу.

Коли мікрооперація, запущена на виконання в пристрої, видаляється з черги планувальника, в ній звільняється місце для прийому нової мікрооперації. Це означає, що ефективний розмір вікна для зміни порядку операцій перевищує довжину цієї черги і обмежується місткістю буфера переупорядкування.

До кожної черги планувальника приєднано одне або декілька спеціалізованих функціональних пристроїв. Число операцій, які можуть бути запущені на виконання в

кожному такті, зазвичай помітно перевищує ширину інших трактів ядра процесора і варіюється від 5 до 10, що дозволяє згладжувати пікове навантаження і забезпечити високу пропускну спроможність при неоднорідному завантаженні пристроїв.

Окрім арифметико-логічних і адресних функціональних пристроїв, в кожному ядру процесора є також пристрої використання і вивантаження (Load/Store), які виконують доступ до кеш-пам'яті даних і до оперативної пам'яті. Ці пристрої працюють асихронно від інших, і їх зазвичай не зображують на блок-схемах. Логічно ці пристрої пов'язані з пристроями обчислення адрес читання/запису (AGU). Пристрої використання і вивантаження конвеєризовані і можуть одночасно обслуговувати велику кількість запитів. Ці пристрої також здійснюють попередню вибірку з оперативної пам'яті (копіювання в кеші тих даних, використання яких очікується найближчим часом).

1.1.2 Структура Haswell процесорного ядра

Структура Haswell ядра приведена на рис. 1.2.

Блок попереджуючої вибірки команд ядра призначений для забезпечення безперебійної вибірки і попереднього декодування інструкцій IA-32/64, для наступного їх декодування в операції мікрокоду. Таким чином, з інструкцій IA-32/64 різної довжини виходить впорядкований потік рівномірних операцій, що іменуються в фірмі Intel "мікрооперaціями" (μops), для наступної обробки із зміною послідовності (out - of - order).

Процес попереднього декодування формує черги з інструкцій IA-32/64 (до шести інструкцій за такт), які завантажуються з кеша L1 в проміжний буфер для наступної передачі на декодування. Від оперативності роботи декодерів, від їх узгодженості з модулем пророцтва галужень і "уміння" заповнювати конвеєр безпосередньо залежить безперебійність потоку мікроопераций, використання конвеєра, і зрештою продуктивність процесора.

Після попереджуючої вибірки і попереднього декодування команди архітектури IA-32/64 подаються на декодери, які, у свою чергу, видають на виході мікрооперації фіксованої довжини для подальшої обробки із зміною послідовності (out - of - order). Три з чотирьох декодерів ядра обробляють прості команди, внаслідок чого кожен видає по одній мікрооперації на виході, тоді як четвертий декодер обробляє складні інструкції і видає до чотирьох мікроопераций. Крім цього, мікропрограмні інструкції розміром більше чотирьох мікроопераций розбиваються на блоки по чотири мікрооперації.

Блоки декодування підтримують як мікро-з’єднання (Micro Fusion), об'єднуючі декілька інструкцій в ряд поодиноких мікрооперацій, так і макро-злиття (Macro Fusion), що об'єднує пари інструкцій в одну мікроінструкцію. У будь-якому випадку, декодери незалежно від характеру інструкцій, що поступають, видають на виході не менше чотирьох мікрооперацій за такт.

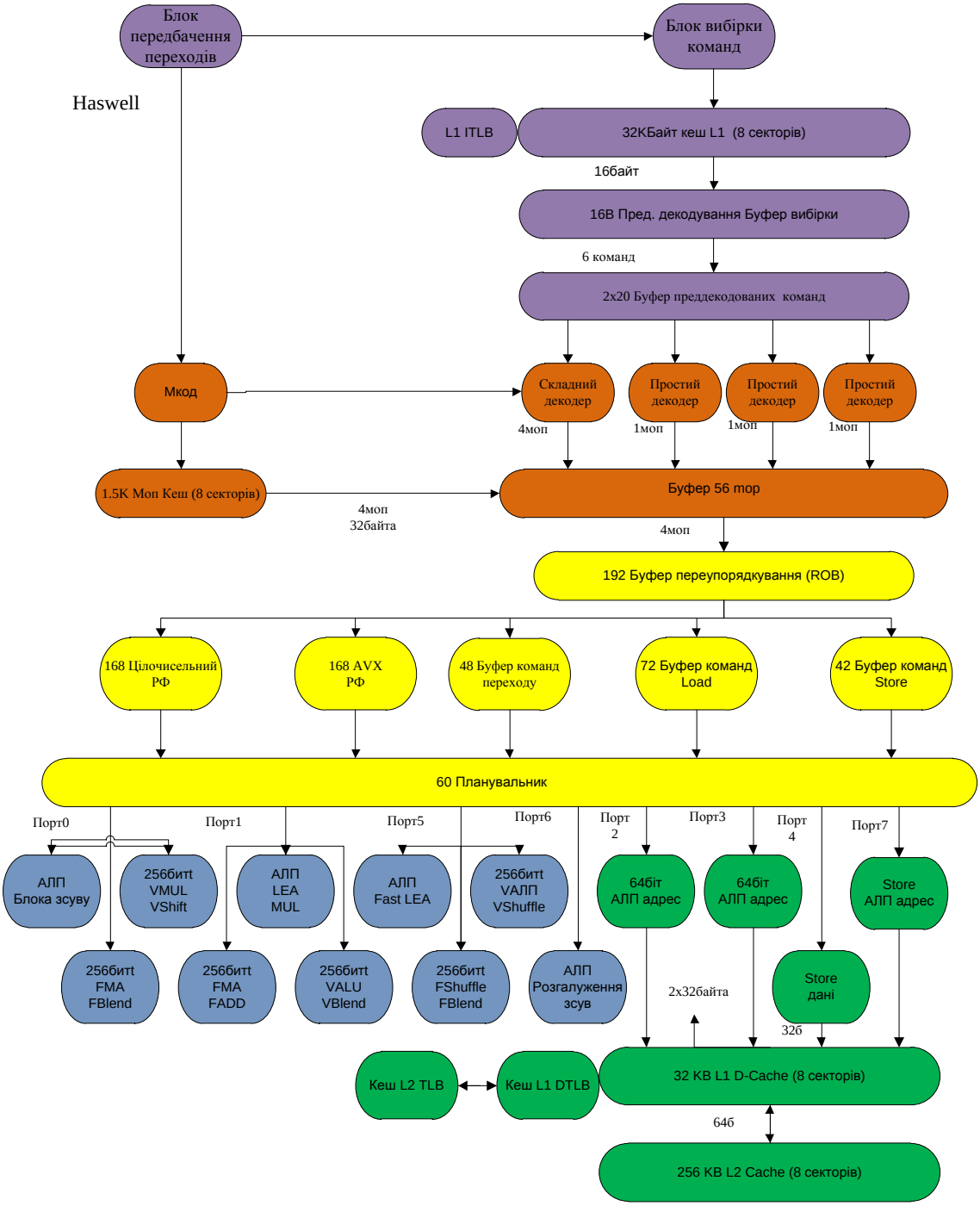


Рис. 1.2 - Структура Haswell ядра процесора фірми Intel

Важливим компонентом структури ядра є кеш декодованих мікрооперацій, чи кеш інструкцій L0. Кеш декодованих мікрооперацій вміщує трохи більше півтори тисяч мікрооперацій. Без особливих затій він кеширує на виході декодерів усі заздалегідь декодовані мікрооперації. Як тільки поступає на обробку нова інструкція, блок попереджуючої вибірки насамперед виробляє звіряння з кешем L0, і в разі виявлення збігів, використання конвеєра по чотири мікрооперації за такт в обхід декодерів здійснюється вже з кеша L0. Незадіяні і простоюючі ланцюги декодерів, до речі, дуже складні, і тому досить "ненажерливі", у цей момент просто відключаються від живлення. Інакше, коли кеш декодованих операцій виявляється незатребуваним, триває звичайна робота по вибірці і декодуванню команд, а кеш декодованих операцій переводиться в режим економії енергії.

Кеш L0 в якійсь мірі можна вважати частиною кеша L1, в який він, до речі, інтегрований, але окремою і дуже швидкою його частиною. За заявами фірми Intel, при роботі з більшістю додатків, вірогідність вдалого "попадання" в кеш декодованих мікрооперацій дуже велика і може досягати 80%.

1.2 Методи дослідження обчислювальних систем

1.2.1 Аналітичні методи

Аналітичні методи дослідження обчислювальних систем зводяться до побудови математичних моделей, які представляють фізичні властивості як математичні об'єкти й відносини між ними, що виражаються за допомогою математичних операцій. Моделі, побудовані цими методами, називаються аналітичними моделями.

При побудові аналітичних моделей властивості об'єктів описуються виходячи із властивостей складових - фізичних елементів або елементарних процесів. Для цього використовується відповідний математичний аналог і за допомогою відповідного математичного апарата будуються вирази, які зв'язують показники, що характеризують елементи.

Як правило, властивості елементів і систем удається представити в аналітичній формі, якщо приймаються певні допущення про властивості й поводження описуваних об'єктів: незалежність одних факторів від інших, лінійність деяких залежностей, миттєвість переходів між станами і т.д. Якщо допущення відповідають реальності, модель добре відтворює залежність між характеристиками й параметрами. Однак у багатьох випадках допущення приводять до істотних відмінностей моделі від реального об'єкта,

внаслідок чого залежність, що моделюється суттєво відрізняється від реальної й характеристики представляються на моделі з великою погрішністю.

Таким чином, аналітичні моделі, базуючись на допущеннях про властивості об'єктів, можуть бути застосовані для дослідження лише тих систем, по відношенню до яких слушні прийняті допущення. Багато систем через специфіку своєї організації недоступні для дослідження аналітичними методами.

Аналітичні методи й моделі розкривають фундаментальні властивості обчислювальних систем і становлять ядро теорії обчислювальних систем.

1.2.2 Чисельні методи

Чисельні методи грунтуються на побудові кінцевої послідовності дій над числами, що приводить до отримання необхідних результатів. При наявності математичної моделі досліджуваного об'єкта застосування чисельних методів зводиться до заміни математичних операцій і відносин відповідними операціями над числами: заміні інтегралів сумами, похідних різницевим відносинами, нескінченних сум кінцевими і т. д. У результаті цього будується алгоритм, що дозволяє точно або з допустимою похибкою визначити значення необхідних величин. Алгоритм реалізується вручну або програмується для ЕОМ. Результат застосування чисельних методів - таблиці (графіки) залежностей, які розкривають властивості об'єкта. Чисельні методи порівняно з аналітичними дозволяють вирішувати значно ширше коло завдань, проте він не підходить для систем з великою кількістю різних станів.

1.2.3 Імітаційні методи

Імітаційні методи засновані на поданні порядку функціонування системи у вигляді алгоритму, який називається імітаційною (алгоритмічною) моделлю. Програма містить процедури, які реєструють стани імітаційної моделі й обробляють зареєстровані дані для оцінки необхідних характеристик процесів і системи, що моделюється.

Імітаційні моделі відтворюють процес функціонування й властивості досліджуваних систем виходячи з апріорно відомих властивостей елементів системи - за рахунок об'єднання моделей елементів у структуру, відповідну до досліджуваної системи, та імітації функціонування елементів у їхній взаємодії.

Дослідження обчислювальних систем імітаційними методами складається з декількох етапів:

1) Визначення принципів побудови моделі. Мета цього етапу - сформувати загальний задум моделі (склад характеристик і параметрів, що підлягають відображенню, область визначення моделі, вимоги до точності результатів моделювання, тип математичної моделі, програмні й технічні засоби для опису й реалізації моделі). На цьому етапі висуваються гіпотези про властивості моделюємої системи, приймаються допущення для використання відповідних математичних методів і конкретизуються експерименти, які будуть проводитися над моделлю;

2) Розробка моделі. Мета цього етапу - створення програми моделювання для ЕОМ. При цьому загальний задум моделі перетворюється в конкретний алгоритмічний опис. Етап завершується перевіркою працездатності й адекватності моделі;

3) Моделювання на комп'ютері. Мета цього етапу - одержання за допомогою моделі даних про функціонування моделі об'єкта , обробка отриманих даних, а при синтезі системи - вибір параметрів, які оптимізують задані характеристики системи й задовольняють заданим обмеженням;

Найважливіша властивість метода імітаційного моделювання - універсальність, що проявляється в наступном. По-перше, метод імітації дозволяє досліджувати системи будь-якого ступеня складності. Ускладнення об'єкта дослідження приводить до збільшення обсягу даних, що вводяться в модель, і часу моделювання на ЕОМ, але при цьому принципи побудови моделей залишаються незмінними. По-друге, метод імітації не обмежує рівень деталізації в моделях. За допомогою алгоритмів можна відтворювати будь-які, скільки завгодно своєрідні взаємозв'язки між елементами системи й процеси функціонування. Більш детальне подання організації й функціонування системи позначається тільки на обсязі на моделях або число експериментів, тобто час моделювання, можна отримати високу точність результатів моделювання. Недоліки імітаційних методів - великі витрати часу на моделювання й приватний характер одержуваних результатів. В імітаційній моделі процес функціонування системи відтворюється у всіх суттєвих для дослідження деталях за рахунок послідовного виконання на ЕОМ операцій над величинами. Число операцій, що забезпечує відтворення представлених інтервалів функціонування системи, виявляється значним і при моделюванні систем помірної складності становить 108-1012 операцій на одну реалізацію моделі. Тому при моделюванні на ЕОМ, котра має швидкодію мільйон операцій у секунду, для одного прогону моделі потрібні хвилини й години процесорного часу. При цьому модель дозволяє оцінити характеристики системи тільки в одній точці, відповідній

до значень параметрів X, уведених у модель перед початком моделювання. Щоб визначити залежність між характеристиками й параметрами, необхідні багаторазові прогони моделі, у результаті яких значення Y визначаються для багатьох наборів параметрів. Можливості методів оптимізації параметрів на імітаційних моделях обмежуються великими витратами часу на моделювання системи в одній точці.

Незважаючи на зазначені недоліки, методи імітаційного моделювання в силу їх універсальності широко використовуються при теоретичних дослідженнях і проектуванні обчислювальних систем. Імітаційні моделі дозволяють дослідникові перед етапу розробки сформувати уявлення про властивості системи й, пізнаючи систему через її модель, ухвалювати обґрунтовані проектні рішення.

1.2.4 Експериментальні методи

Експериментальні методи ґрунтуються на одержанні даних про функціонування обчислювальних систем у реальних або спеціально створених умовах з метою оцінки якості функціонування й виявлення залежностей, що характеризують властивості систем і їх складових. Типові задачі, котрі розв'язуються експериментальними методами, - оцінка продуктивності й надійності системи, визначення складу й кількісних показників системного навантаження залежно від прикладного навантаження і т.д.

Експериментальні дослідження проводяться в наступному порядку:

1) Формулюється мета дослідження;алгоритмічного опису моделі (програми) і витратах часу на моделювання. Особливості організації й функціонування, що перешкоджають використанню аналітичних методів, легко відтворюються в імітаційних моделях. По-третє, імітаційна модель є необмеженим джерелом даних про поводження досліджуваної системи - нові експерименти на моделі дозволяють отримувати додаткові дані про систему. За рахунок цього гарантується детальна оцінка характеристик, функціонування як системи в цілому, так і її складових. Як правило, збільшуючи тривалість експериментів.

2) Обирається або розробляється методика дослідження, яка встановлює модель досліджуваного об’єкту; спосіб і засоби вимірювання; спосіб і засоби вимірювальних даних, а також інтерпретація вимірювальних даних;

3) Проводяться вимірювання процесу функціонування об’єкту в реальних або спеціально створюваних умовах;

4) Вимірювальні дані обробляються й відповідним чином інтерпретуються.

Експериментальні методи забезпечують одержання найбільш достовірних даних про обчислювальні системи, у чому й полягає їхня перевага в порівнянні з аналітичними й імітаційними методами, заснованими на використанні моделей. У багатьох випадках експериментальні методи є єдиним джерелом інформації про функціонування й властивості обчислювальних систем.

Недоліки експериментальних методів - великі витрати праці й часу на проведення експериментальних досліджень, а також приватний характер одержуваних результатів, поширення яких на системи з іншою конфігурацією й режимом функціонування вимагає досить складної роботи.

1.2.5 Вибір методу дослідження об’єкту

В роботі буде досліджуватися ефективність конвеєрного ядра комп’ютера в залежності від його параметрів та структури.

При дослідженні складних систем аналітичним методом вводиться значна кількість припущень, тому застосування його потребує прискіпливої перевірки точності отриманих результатів. Використання цього методу дуже привабливе, але досліджуваній системі властиве велике число різних станів, яке складно (а інколи і неможливо) описати графічно, отже, застосування чисельних методів також обмежено.

Для проведення дослідження експериментальним методом необхідна наявність фізичної моделі досліджуваної системи, що в даному випадку робить цей метод незастосовним. Оптимальним варіантом є імітаційний метод, який не вимагає великих тимчасових витрат, і дає можливість дослідження системи за допомогою зміни її характеристик.

Таким чином, дослідження ефективності функціонування проводитиметься де можливо аналітичними та чисельними методами, а де неможливо - імітаційними методом.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЯДЕР КОМП'ЮТЕРІВ

2.1 Дослідження ефективності ядра з універсальним функціональним пристроєм

2.1.1 Модель ядра в безперервному часі

Будь-яка модель об'єкту - це спрощене представлення основних особливостей цього об'єкту. Модель має бути максимально простою, щоб забезпечити її дослідження аналітичними методами, і при цьому відображати основні, істотні для дослідження риси об'єкту.

В розробленій моделі має бути збережена така особливість структури ядра як конвейерність. В реальних ядрах кількість етапів 10 і більше. В підсистемі підготовки команд для виконання (пристрої керування - ПК), яка читає команди, дешифрує їх, готує до виконання - не менше 5-6 етапів.

У виконавчій частині (підсистемі виконання команд), яка складається з багатьох спеціалізованих функціональних пристроїв, також досить велика кількість етапів конвеєра.

Між всіма етапами (стадіями) конвеєра є буфери різного обсягу.

Як відомо, продуктивність (пропускна спроможність) конвеєра визначається продуктивністю його найповільнішої ланки. Після заповнення конвеєра саме з такою продуктивністю (пропускною спроможністтю) і видаватимуться команди на виконання в підсистему виконання команд.

В ідеальному конвеєрі підготовки команд тривалість всіх етапів однакова. В реальних конвеєрах підготовки команд тривалісті окремих етапів (наприклад, читання команди з підсистеми пам'яті, коли команди немає в підсистемі кеш-пам'яті) не рівні. Саме такі ситуації і обумовлюють випадковий характер підготовки команд для виконання.

Для цілей дослідження всі етапи конвеєра підсистеми підготовки команд (ПК) доцільно замінити одним ступенем з еквівалентною продуктивністю.

Аналогічно всі етапи підсистеми виконання команд також замінюються одним ступенем з еквівалентною продуктивністю. Для цілей дослідження інтегральних характеристик ядра процесора байдуже чи являється підсистема виконання універсальним функціональним пристроєм (ФП), чи складається з декількох

спеціалізованих функціональних пристроїв. При цьому не має значення чи виконувалися команди в спеціалізованих функціональних пристроях строго по порядку, чи ні (позачергове виконання команд), оскільки результати виконання команд записуються в архітектурні регістри строго в порядку передбаченому програмою, тобто послідовно в порядку номерів команд програми. Отже, структура з спеціалізованими функціональними пристроями замінена на структуру з універсальним функціональним пристроєм з еквівалентною продуктивністю. Також для цілей дослідження не має значення як реалізований універсальний функціональний пристрій - конвеєрний чи комбінаційний.

Передбачається, що в послідовності команд виконуваної ядром програми відсутні які-небудь команди, які створюють "перешкоди" роботі конвеєра. Після заповнення конвеєра підготовки команд він з деякою пропускною спроможністю генерує команди (заявки) для підсистеми виконання команд.

Спрощена структура моделі ядра представлена на рис. 2.1.

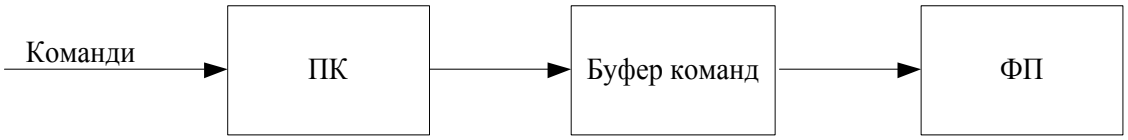


Рис. 2.1 - Спрощена структурна схема моделі ядра

ПК - пристрій керування, який генерує команди.

Буфер команд - накопичує команди в разі неготовності ФП до їх обробки.

ФП - функціональний пристрій, який виконує команди.

Тоді процес функціонування ядра з двоступінчатим конвеєром можна представити таким чином.

На вхід ядра поступає нескінченна послідовність команд виконуваної програми. Передбачається, що всі команди програми є командами обробки різного типу (обробки цілих чисел, чисел з плаваючою комою і тому подібне). Поява команд обробки різного типу в програмі випадкова. Закон розподілу команд в програмі - рівномірний з вірогідністю ωі, а $\sum_{i=1}^{n} \omega_i = 1$.

Модель ядра складається з двох східців. Перший ступінь - пристрій керування (ПК), другий ступінь - функціональні пристрої (ФПі) для виконання операцій і -го типу.

ПК читає чергову команду з пам'яті. Якщо чергова команда знаходиться в кеш команд, то вона швидко (одиниці тактів) доставляється в регістр команди. Якщо ж

звернення в кеш команд невдале, то виконується читання рядка кеш з оперативної пам'яті та передача необхідної команди в регістр команди (час значно більший, ніж при вдалому зверненні). За відсутності сторінки в оперативній пам'яті формується запит на переривання. ПК припиняє виконувати чергову команду, виконує підкачування відсутньої сторінки з диска в оперативну пам'ять, коригує необхідні таблиці і повертається до виконання перерваної команди. Рядок з оперативної пам'яті переноситься в кеш команд, а необхідна команда записується в регістр команди.

Після читання чергової команди ПК дешифрує її. Якщо команда відноситься до групи команд $Ri*Rj \Rightarrow Rk$ (обидва операнди і результат операції знаходяться в регістрах процесора), то команда передається на виконання у відповідний ФПі (з вірогідністю ωі - якщо команда і -го типу). Якщо ж команда відноситься до групи $Ri*П \Rightarrow Rk$ (один з операндів знаходиться в пам'яті), то формується адреса операнда відповідно до способу адресації і видається запит до підсистеми пам'яті.

Якщо необхідний операнд знаходиться в кеш даних, то він швидко доставляється в регістр, і команда передається на виконання у відповідний ФПі. Якщо ж звернення в кеш даних невдале, то виконується читання рядка кеш з оперативної пам'яті і передача необхідного операнда в ПК і далі у відповідний ФПі (час значно більший, ніж при вдалому зверненні). За відсутності сторінки в оперативній пам'яті формується запит на переривання. ПК припиняє виконувати чергову команду, виконує підкачування відсутньої сторінки з диска в оперативну пам'ять, коригує необхідні таблиці і повертається до виконання перерваної команди. Рядок з оперативної пам'яті записується в кеш даних, необхідний операнд записується в ПК, а команда відповідно до коду операції видається у відповідний ФПі.

Процес генерування чергової команди в ПК є випадковим. Тривалість підготовки чергової команди залежить від багатьох чинників, таких як: тип команди (до якої групи команд вона відноситься), вірогідність вдалого звернення до підсистеми пам'яті за командою, довжина команди, спосіб адресації операнда в пам'яті (якщо він знаходиться в пам'яті), вірогідність вдалого звернення в підсистему пам'яті за операндом.

Приймається, що потік згенерованих ПК команд це найпростіший потік, а тривалість підготовки чергової команди ПК залежить тільки від архітектурних і структурних особливостей ядра, але не залежить ні від типу команди (першого або другого), ні від, наприклад, кода операції - складання, віднімання, множення і тому подібне.

Якщо ФПі вільний, то чергова готова до виконання команда і-го типу з ПК відразу поступає на обробку у ФПі. Якщо ж ФПі зайнятий виконанням раніше згенерованих ПК команд, то чергова команда з ПК поступає в буфер команд ФПі. Якщо буфер команд ФПі заповнений, то ПК припиняє генерацію (блокується) до появи вільного місця в буфері команд ФПі (тобто до виконання чергової команди ФПі). ФПі виконує обробку операндів відповідно до коду операції команди. Команди вибираються або безпосередньо з ПК, або з буфера команд ФПі. Тривалість виконання чергової команди у ФПі є випадковою із-за дії таких чинників:

команди розрізняються, передусім, за типом операндів, а саме: для ФПі це може бути обробка цілих чисел або логічних операндів; для ФПі з плаваючою точкою – короткий формат або довгий формат. Тривалісті виконання операцій можуть істотно відрізнятися.

часи виконання команд усередині одного типу операндів також можуть істотно відрізнятися (наприклад, складання і множення, і тому подібне);

як і при розгляді процесу генерації команд в ПК приймається, що потік виконаних у ФПі операцій найпростіший.

Команди з буфера команд вибираються на виконання у ФПі згідно з дисципліною FIFO. Якщо в буфері немає готових до обробки команд і ПК не згенерував у цей момент часу чергову команду, то ФПі простоює.

Введемо поняття:

i - кількість команд в моделі (як згенерованих,кі знаходяться в буфері, так і на виконанні в ФП);

λ - інтенсивність генерації команд

$$\lambda = \frac{1}{Тген.}. \qquad (2.1)$$

Тген. – математичне очікування часу генерації команди;

μ - інтенсивність виконання команд

$$\mu = \frac{1}{Твик.}. \qquad (2.2)$$

Твик – математичне очікування часу виконання команди.

Якщо $\lambda = \mu$, то система є збалансованою. Якщо $\lambda > \mu$, то команди не встигають пройти обробку і накопичуються в буфері розмірністю **n** . Якщо $\lambda < \mu$, то ФП встигає виконувати команди і частину часу простоює. Буфер в цьому випадку заповнюється не повністю.

Досліджувана система може перебувати в таких станах:

1) a_0 - ПК генерує чергову заявку, буфер - порожній, ФП простоює;

- a_i ($1 \le i \le n$) - ПК генерує чергову заявку, в буфері (i - 1) заявка, ФП виконує одну заявку.

- a_{n+1} - ПК блокований, в буфері n заявок (повністю заповнений), ФП виконує одну заявку.

В кожному з цих станів система може знаходитися з вірогідністю $P_0, P_1, P_2, ...P_{n+1}$.

$$\sum_{n=0}^{n+1} P_i = 1. \tag{2.3}$$

Граф станів моделі приведений на рис. 2.2

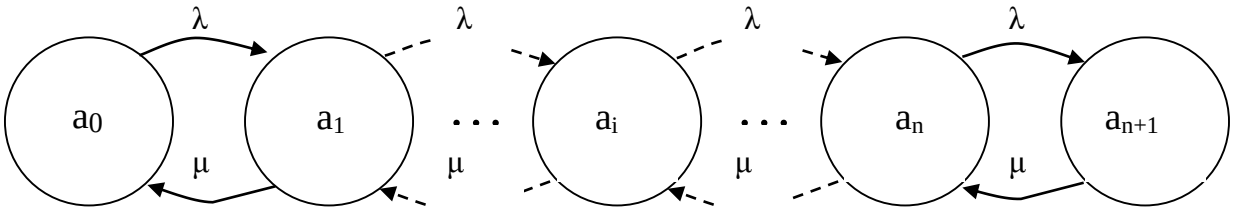


Рис. 2.2 - Граф станів моделі

2.1.2 Система рівнянь

По графу складається система рівнянь для сталого режиму. Похідні за часом від вірогідності станів дорівнюватимуть нулю, і система рівнянь виглядатиме таким чином.

$$\begin{cases} P_{0*}\lambda = P_{1*}\mu \\ P_{i*}(\lambda+\mu) = P_{i-1*}\lambda + P_{i+1*}\mu, \ 1 \le i \le n \\ P_{n+1*}\mu = P_{n*}\lambda \end{cases} \tag{2.4}$$

Система рівнянь містить **n+2** невідомих при такій же кількості рівнянь.

Замінимо одне з рівнянь на $\sum_{i=0}^{n+1} P_i = 1$.

Перетворимо систему рівнянь, розділивши всі складові на μ, і позначимо через $\rho=\lambda/\mu$ коефіцієнт навантаження ФП.

Отримаємо наступну систему рівнянь.

$$\begin{cases} P_0 * \rho = P_1 \\ P_{i*}(\rho + 1) = \rho * P_{i-1} + P_{i+1} \; ; \quad 1 \le i \le n \\ P_{n+1} = \rho * P_n \end{cases} \tag{2.5}$$

Виразимо всі Pi через P0. Отримаємо

$$\begin{cases} P_1 = P_0 * \rho \\ P_i = P_0 * \rho^i \qquad 1 \le i \le n \\ P_{n+1} = P_0 * \rho^{n+1} \end{cases} \tag{2.6}$$

Підставивши значення P_i у формулу $\sum_{i=0}^{n+1} P_i = 1$ визначається

$$P_0 = \frac{1}{\rho^0 + \rho^1 + \rho^2 + ... + \rho^{n+1}}. \tag{2.7}$$

Неважко помітити, що в знаменнику $\rho^0+\rho^1+\rho^2+\ldots+\rho^{n+1}$ - це геометрична прогресія, яка складається з $n+2$ елементів, де перший елемент дорівнює 1. Сума **n+2** членів геометричної прогресії, при $\rho \neq 1$, буде

$$S_{n+2} = \frac{1-\rho^{n+2}}{1-\rho}. \tag{2.8}$$

Тоді

$$P_0 = \frac{1-\rho}{1-\rho^{n+2}}, \tag{2.9}$$

$$P_i = \rho^i * P_0 = \rho^i * \frac{1-\rho}{1-\rho^{n+2}}, \quad 1 \le i \le n+1. \tag{2.10}$$

Оцінками ефективності функціонування системи будуть такі показники:

- коефіцієнт використання ПК;
- коефіцієнт використання ФП;

Параметрами є:

- інтенсивність генерації команд λ;
- інтенсивність виконання команд μ;
- розмір буфера n.

Коефіцієнт використання ФП:

$$H_{ФП} = 1 - P_0 = 1 - \frac{1-\rho}{1-\rho^{n+2}} = \frac{\rho(1-\rho^{n+1})}{1-\rho^{n+2}} \quad \text{- для } \rho \neq 1. \tag{2.11}$$

Вираз для $H_{ФП}$ при $\rho = 1$ знайдемо, застосувавши, наприклад, правило Лопіталя.

$$H_{ФП} = \frac{n+1}{n+2} \text{ - для } \rho = 1. \quad (2.12)$$

Коефіцієнт використання ПК:

$$E_{ПК} = 1 - P_{n+1} = \frac{1-\rho^{n+1}}{1-\rho^{n+2}} \quad \text{- для } \rho \neq 1. \quad (2.13)$$

Вираз для $E_{ПК}$ при ρ = 1 знайдемо, застосувавши, як і раніше, правило Лопіталя.

$$E_{ПК} = \frac{n+1}{n+2} \text{ - для } \rho = 1. \quad (2.14)$$

Отже, вирази для показників ефективності досліджуваної моделі такі:

коефіцієнт використання ФП

$$H_{ФП} = \frac{\rho(1-\rho^{n+1})}{1-\rho^{n+2}} \text{ - для } \rho \neq 1, \quad (2.15)$$

$$H_{ФП} = \frac{n+1}{n+2} \text{ - для } \rho = 1. \quad (2.16)$$

коефіцієнт використання ПК

$$E_{ПК} = \frac{1-\rho^{n+1}}{1-\rho^{n+2}} \text{ - для } \rho \neq 1, \quad (2.17)$$

$$E_{ПК} = \frac{n+1}{n+2} \text{ - для } \rho = 1. \quad (2.18)$$

2.1.3 Дослідження показників ефективності моделі

Коефіцієнти використання ПК і ФП є функціями двох змінних - ρ і n. Дослідження функцій двох змінних проводиться шляхом фіксації чергового значення однієї із змінних і зміни в певних межах іншої змінної.

Необхідно зробити вибір, яка ж з двох змінних змінюється в меншому діапазоні значень. Саме значення цієї змінної доцільно по черзі фіксувати і змінювати при цьому значення іншої змінної.

Коефіцієнт навантаження ρ якісно може бути:

$\rho = 1$ - це відповідає збалансованому режиму роботи моделі, тобто ситуації, коли інтенсивність генерації заявок і інтенсивність їх виконання однакові.

$\rho \neq 1$ - це відповідає незбалансованому режиму роботи моделі, тобто ситуації, коли інтенсивність генерації заявок і інтенсивність їх виконання різні.

Виходячи з цього, розглядається функціонування моделі в трьох режимах:

- збалансована система - $\rho = 1$;

- не збалансована система - $\rho > 1$;

- не збалансована система - $\rho < 1$.

Дослідження поведінки системи в не збалансованих режимах проводиться при яскраво вираженій незбалансованості, наприклад, при $\rho = 2$ і $\rho = 0,5$.

При аналізі коефіцієнтів використання ПК і ФП встановлювалося:

- як вони змінювалися в різних режимах;

- які розміри буферів доцільно рекомендувати для досягнення їх максимально можливих значень.

Розміри буферів змінювалися в широких межах - $n = 0,1,2,4,8,16,32,64$.

Точність аналітичних результатів перевірялася за допомогою чисельних експериментів. Результати аналітичних результатів та чисельних експериментів відображені в таблицях 2.1 – 2.3.

2.1.3.1 Збалансована система ($\rho = 1$)

Показники ефективності (значення коефіцієнтів використання пристроїв) як функції розміру буфера при $\rho = 1$ приведені в таблиці 2.1.

Таблиця 2.1 - Показники ефективності (значення коефіцієнтів використання пристроїв) як функції розміру буфера при $\rho = 1$

n		2	4	8	16	32	64
H	Аналітичні результати	0.750	0.833	0.900	0.944	0.971	0.985
E		0.750	0.833	0.900	0.944	0.971	0.985
H	Чисельний експеримент	0.75	0.834	0.9	0.945	0.970	0.988
E		0.75	0.834	0.9	0.944	0.971	0.985

При n = 16 і більше коефіцієнти використання пристроїв (як ФП, так і ПК) близькі до своїх граничних значень, рівних одиниці. Можна рекомендувати вибирати буфер не більше 32. Відмінність коефіцієнтів використання від граничних значень складе в цьому випадку не більше 3%.

2.1.3.2 Не збалансована система ($\rho > 1$)

Показники ефективності як функції розміру буфера при $\rho = 2$ приведені в таблиці 2.2.

Таблиця 2.2 - Показники ефективності як функції розміру буфера при $\rho = 2$

n		1	2	4	8	16	32
H	Аналітичні результати	0.857	0.933	0.984	0.999	1	1
E		0.429	0.467	0.492	0.4995	0,5	0.5
H	Чисельний експеримент	0.938	0.975	0.995	1	1	1
E		0.429	0.467	0.492	0.499	0.5	0.5

При $\rho > 1$ коефіцієнт використання менш продуктивного пристрою моделі (а ним є ФП) швидко прагне до свого граничного значення, рівного 1 (тим швидше, чим більш система незбалансована). Так, для $\rho = 2$ при $n = 4$ коефіцієнт використання ФП вже практично рівний 1, а при $\rho = 1{,}1$ близький до граничного значення при $n = 8$.

Поведінка коефіцієнта використання більш продуктивного компоненту моделі ПК залежно від розміру буфера при різних ρ аналогічна поведінці коефіцієнта використання ФП, тільки граничне значення буде $1/\rho$.

Таким чином, вибравши буфер $n \geq 8$, в системі будуть забезпечені коефіцієнти використання пристроїв, близькі до граничних значень, рівних 1 - для менш продуктивного компоненту системи (ФП) і $1/\rho$ - для продуктивнішого (ПК).

2.1.3.3 Не збалансована система ($\rho < 1$)

Показники ефективності як функції розміру буфера при $\rho = 0{,}5$ приведені в таблиці 2.3.

Таблиця 2.3 - Показники ефективності як функції розміру буфера при $\rho = 0,5$

n		1	2	4	8	16	32
H	Аналітичні результати	0.429	0.467	0.492	0.4995	0.5	0.5
E		0,857	0,933	0,984	0,9990	1	1
H	Чисельний експеримент	0.429	0.467	0.492	0.4995	0.541	0.556
E		0.857	0.933	0.984	0.999	1	1

При $\rho < 1$ коефіцієнт використання менш продуктивного пристрою моделі (а ним є ПК) швидко прагне до свого граничного значення, рівного 1 (тим швидше, чим більш система незбалансована). Так, при $\rho = 0,5$ і $n = 4$ коефіцієнт використання ПК вже практично дорівнює 1, а при $\rho = 0,9$ - близький до граничного значення при $n = 8$.

В разі детермінованих часів генерації та виконання команд буфер заявок не потрібний, а значення коефіцієнтів використання пристроїв будуть:

1 - для збалансованої моделі при $\rho = 1$;

В не збалансованому режимі при $\rho > 1$:

1 - коефіцієнт використання ФП (менш продуктивного пристрою моделі);

$1/\rho$ - - коефіцієнт використання ПК (продуктивнішого елементу моделі);

в не збалансованому режимі при $\rho < 1$:

1 - коефіцієнт використання ПК (менш продуктивного пристрою моделі);

ρ - коефіцієнт використання ФП (продуктивнішого елементу моделі).

2.2 Дослідження ефективності функціонування комп'ютера з одноядерним процесором з m спеціалізованими функціональними пристроями

В реальних високопродуктивних ядрах сучасних процесорів підсистема обробки складається з багатьох спеціалізованих функціональних пристроїв (наприклад, з: ФП для складання/віднімання чисел з фіксованою точкою, ФП для множення чисел з фіксованою точкою, ФП для ділення чисел з фіксованою точкою, універсальни ФП для виконання команд зрушень, ФП для складання/віднімання чисел з плаваючою точкою, дійсних чисел), ФП для множення дійсних чисел, ФП для ділення дійсних чисел, кеш-пам'яті даних першого рівня, кеш-пам'яті другого рівня, кеш-пам'яті третього рівня, контролера оперативної пам'яті). Було показано, що для цілей дослідження загальної

ефективності компонентів ядра процесора та спрощення моделі може бути здійснена заміна спеціалізованих ФП одним універсальним ФП з еквівалентною продуктивністю.

Була розроблена методика дослідження, отримані інтегральні оцінки ефективності узагальнених пристрої керуваня і універсального функціонального пристрою (підсистеми виконання команд).

Враховуючи, що підсистема виконання команд сучасних високопродуктивних ядер процесорів складається з багатьох спеціалізованих функціональних пристроїв, необхідно мати не лише інтегральну (узагальнену) оцінку ефективності всієї підсистеми виконання команд, але і оцінки ефективності для кожного спеціалізованого функціонального пристрою в підсистеми виконання команд ядра процесора при виконанні різних програм.

2.2.1 Модель суперскалярного ядра з m ФП та індивідуальними буферами заявок

В ядрах сучасних процесорів можливі такі варіанти структур буферів згенерованих команд:

- спільний буфер;
- індивідуальні буфери для групи функціональних пристроїв;
- індивідуальні буфери для кожного функціонального пристрою.

В даном розділі буде досліджена модель ядра з індивідуальними буферами для кожного функціонального пристрою.

Як і раныше вважатимемо, що модель ядра складається з двох етапів (ступенів). Перший етап це підсистема підготовки команд, другий етап це m ФП з індивідуальними буферами заявок.

Модель ядра функціонує таким чином.

На вхід пристрою керуваня поступає нескінченна послідовність команд виконуваної програми. Передбачається, що програма складається з команд обробки різного типу (обробки цілих чисел, чисел з плаваючою точкою і тому подібне), команд читання, команд запису. Поява команд різного типу в програмі випадкова. Закон розподілу команд в програмі - рівномірний з вірогідністю Wi, а $\sum_{i=1}^{n} W_i = 1$.

Чергова згенерована команда i-го типу (заявка) або передається на виконання в один з $ФП_i$, (з вірогідністю Wi) якщо він вільний, або записується в індивідуальний

буфер заявок (команд) пристрою, якщо $ФП_i$ зайнятий виконанням попередньої команди і в буфері є вільне місце.

Якщо індивідуальний буфер заявок заповнений, то ПК припиняє генерацію команд (блокується) до тих пір, поки в цьому буфері не з'явиться вільне місце для згенерованої заявки.

Передбачається, що потік згенерованих ПК команд найпростіший з показовим законом розподілу часу між сгенерированними командами. Інтенсивність генерації команд λ ($\lambda = 1/t_{ген.}$, де $t_{ген}$ - математичне очікування часу генерації команди).

Передбачається також, що час генерації чергової команди ПК залежить тільки від архітектурних і структурних особливостей ядра процесора, але не залежить, наприклад, від типу операції - складання, віднімання, множення і тому подібне.

"Перешкоди" роботі конвеєра в пристрої керуваня відсутні.

Допущення про відсутність "перешкод" роботі конвеєра підготовки команд (як і в попередньому розділі) дозволяє значно спростити модель ядра. Отримані при дослідженні такої моделі оцінки ефективності це верхня межа значень ефективності. Реальні оцінки ефективності будуть меншими.

Функціональні пристрої можуть бути як комбінаційного, так і конвеєрного типу.

Комбінаційний ФП це такий пристрій, який залишається зайнятим на весь час виконання однієї операції. Нова заявка приймається на виконання тільки після завершення попередньої заявки.

Конвеєрний ФП це пристрій, який розбитий на ряд етапі (ступенів), починаючи з першого і закінчуючи останнім (к-м). Приймається, що тривалість дій кожного ступеня однакові. Чергова заявка поступає на вхід першого ступеня. Після виконання необхідних дій результат передається на вхід другого ступеня. Після видачі результату другому ступеню перший ступінь може починати виконувати свої дії над черговою заявкою. Аналогічно працюють і інші ступені. Заявка вважається виконаною, коли вона покидає останній ступінь конвеєрного ФП. Пропускна спроможність конвеєрного ФП дорівнює пропускній спроможності його найповільнішого ступеня

Досліджувані структури ядра процесора можуть складатися з таких ФП:

Функціональний пристрій для виконання операцій складання (віднімання) чисел з фіксованою точкою комбінаційного типу з часом виконання операції 1 такт (час виконання детермінована величина).

Функціональний пристрій для виконання операцій множення чисел з фіксованою точкою конвеєрного типу з часом спрацьовування ступеня 1 такт (час виконання детермінована величина).

Функціональний пристрій для виконання операцій ділення чисел з фіксованою точкою комбінаційного типу з часом виконання операції близько 20 тактів (час виконання детермінована величина).

Функціональний пристрій для виконання універсальних операцій над числами з фіксованою точкою (наприклад, різні арифметичні зрушення), логічних операцій комбінаційного типу (час виконання випадкова величина з показовим законом розподілу).

Функціональний пристрій для виконання операцій складання (віднімання) чисел з плаваючою точкою конвеєрного типу з часом спрацьовування ступеня 1 такт (час виконання детермінована величина).

Функціональний пристрій для виконання операцій множення чисел з плаваючою точкою конвеєрного типу з часом спрацьовування ступеня 1 такт (час виконання детермінована величина).

Функціональний пристрій для виконання операцій ділення чисел з плаваючою точкою комбінаційного типу (час виконання детермінована величина).

Функціональний пристрій для виконання команд звернення (читання LD і запису ST) в підсистему пам'яті комбінаційного типу. Він приступає до виконання наступної заявки після завершення обслуговування попередньої заявки.

Якщо $ФП_i$ вільний і в його буфері немає заявок і-го типу, то чергова згенерована пристроєм керування (підсистемою підготовки команд для виконання) заявка і-го типу відразу поступає на виконання в $ФП_i$.

Якщо ж $ФП_i$ зайнятий виконанням раніше згенерованих ПК команд і-го типу, то чергова команда і-го типу з ПК заноситься в індивідуальний буфер заявок (команд) функціонального пристрою $ФП_i$.

Якщо $ФП_i$ закінчив виконання чергової заявки і в його буфері є хоч би одна заявка і-го типу, то вона поступає на виконання в $ФП_i$.

Якщо $ФП_i$ закінчив виконання чергової заявки і в його буфері немає заявок і-го типу, то $ФП_i$ простоює.

Приймається, що потік виконаних в $ФП_i$ заявок (операцій) найпростіший, а час виконання заявки в $ФП_i$ це випадкова величина, розподілена за показовим законом з

інтенсивністю μ_i. Команди з буфера команд вибираються на виконання в $ФП_i$ згідно з дисципліною FIFO.

Структура моделі ядра, яка складається з ПК, m ФП з індивідуальними буферами заявок, представлена на рис.2.1.

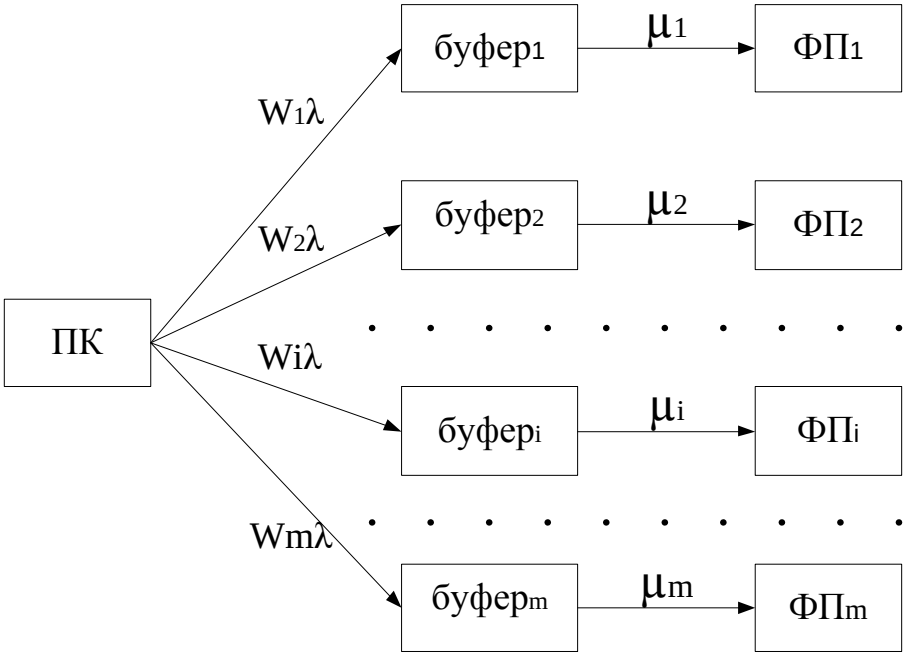


Рис. 2.1 - Структура моделі ядра з індивідуальними буферами заявок для ФП.

$W_i\lambda$ - інтенсивність генерації команд і-го типу; μ_i - інтенсивність виконання команд і-го типу $ФП_i$.

2.2.2. Модель суперскалярного ядра з 2 ФП та індивідуальними буферами заявок

Структура моделі ядра з 2-ма спеціалізованими функціональними пристроями, приведена на рис. 2.2.

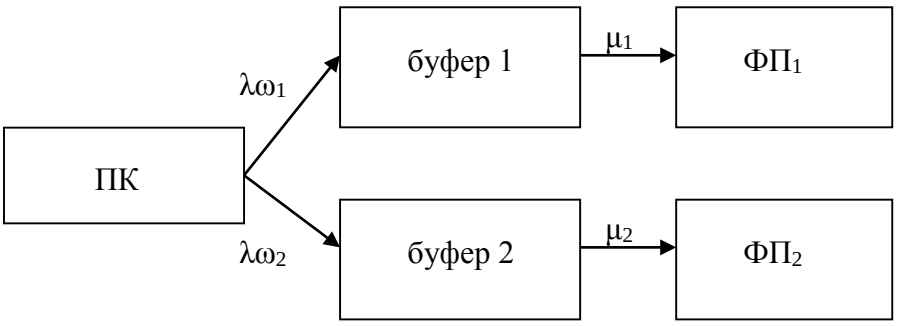


Рис. 2.2 - Структура моделі ядра з 2-ма спеціалізованими функціональними пристроями ($\lambda_1 = W_1\lambda$; $\lambda_2 = W_2\lambda$)

Модель може перебувати в таких станах:

- $a_{0,0}$ - ПК генерує першу заявку, буфери пусті, ФП1, ФП2 - простоюють;

- $a_{i,0}$ ($1 \le i \le n$) - ПК генерує наступну заявку, в буфері 1 (i-1) заявка, буфер 2 порожній, ФП1 – виконує одну заявку, ФП2 - простоює. Для виконання поступають тільки команди 1-го типу.

- $a_{0,j}$ ($1 \le i \le n$) - ПК генерує наступну заявку, в буфері 1 - 0 заявок, в буфері 2 - (j-1) заявка, ФП1, ФП2 – виконують по одній заявці. Для виконання поступають тільки команди 2-го типу.

- $a_{i,j}$ ($1 \le i \le n$) - ПК генерує наступну заявку, в буфері 1 (i-1) заявка, в буфері 2 (j-1) заявка, ФП1 і ФП2 – виконують по одній заявці. Для виконання поступають команди як 1-го так і 2-го типу.

- $a_{n+1,0}$ - ПК блокований, в буфері 1 **n** заявок, в буфері 2 0 заявок, ФП1 – виконує одну команду 1-го типу, ФП2 - простоює;

- $a_{0,n+1}$ - ПК блокований, в буфері 1 0 заявок, в буфері 2 **n** заявок, ФП1 - простоює, ФП2 - виконує одну заявку 2-го типу;

- $a_{n+1,j}$ - ПК блокований, в буфері 1 **n** заявок, в буфері 2 (j-1) заявка, ФП1 і ФП2 – виконують по одній заявці. Для виконання поступають команди як 1-го так і 2-го типу відповідно.

- $a_{i,n+1}$ - ПК блокований, в буфері 1 (i-1) заявка, в буфері 2 **n** заявок, ФП1 і ФП2 – виконують по одній заявці. Для виконання поступають команди як 1-го так і 2-го типу відповідно.

В кожному з таких станів система може знаходитися з вірогідністю $P_{i,j}$, а $\sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij} = 1$.

Граф стану моделі представлений на рис.2.3.

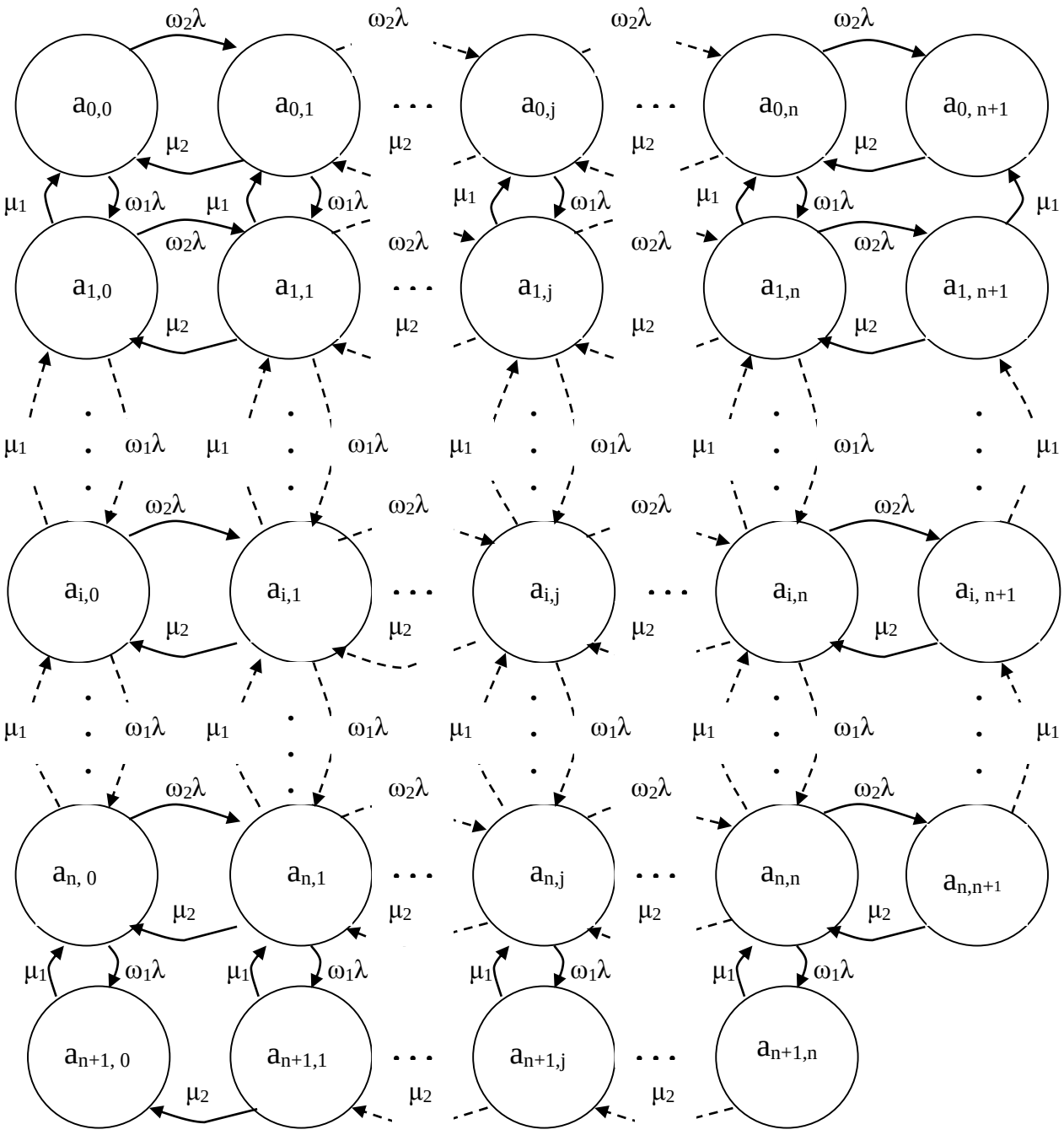


Рис. 2.3 - Граф стану моделі ядра з 2-мя спеціалізованими функціональними пристроями

2.2.3 Система рівнянь

Система рівнянь для вірогідності станів буде такою:

$$P_{00} * \lambda = P_{10} * \mu_1 + P_{01} * \mu_2, \qquad (2.19)$$

$$P_{i0} * (\lambda + \mu_1) = P_{i+10} * \mu_1 + P_{i1} * \mu_2 + P_{i-1.0} * \omega_1 * \lambda \ ; \quad i = 1, 2, ...n+1, \qquad (2.20)$$

$$P_{0j} * (\lambda + \mu_2) = P_{0.j+1} * \mu_2 + P_{1.j} * \mu_1 + P_{0.j-1} * \omega_2 * \lambda; \quad j = 1,2,.....n+1 \qquad (2.21)$$

.....

$$P_{i.j} * (\mu_1 + \mu_2 + \lambda) = P_{i+1.j} * \mu_1 + P_{i.j+1} * \mu_2 + P_{i.j-1} * \omega_2 * \lambda + P_{i-1.j} * \omega_1 * \lambda, \quad (2.22)$$

$$i = 1,2,.....n+1; \quad j = 1,2,.....n+1$$

.....

$$P_{n+1.0} * \mu_1 = P_{n.0} * \omega_1 * \lambda, \qquad (2.23)$$

$$P_{0.n+1} * \mu_2 = P_{0.n} * \omega_2 * \lambda, \qquad (2.24)$$

$$P_{n,n+1} * (\mu_1 + \mu_2) = P_{n,n} * \omega_2 * \lambda_2 , \qquad (2.25)$$

$$P_{n,n+1} * (\mu_1 + \mu_2) = P_{n,n} * \omega_2 * \lambda_1 , \qquad (2.26)$$

$$P_{n+1,j} * (\mu_1 + \mu_2 +) = P_{n,j} * \omega_1 * \lambda + P_{n+1,j+1} \cdot \mu_2 ; \quad j = 1,2,.....n \qquad (2.27)$$

$$P_{i,n+1} * (\mu_1 + \mu_2) = P_{i,n} * \omega_2 * \lambda + P_{i+1,n+1} * \mu_1; \qquad i = 1,2,.....n . \qquad (2.28)$$

В загальному вигляді система рівнянь для вірогідності станів моделі складається з $(n+2)^2$ -1 рівняннь і такої ж кількості невідомих. Замінивши будь-яке рівняння на рівняння $\sum_{i=0}^{n+1}\sum_{j=0}^{n+1} P_i j = 1$, можна вирішити цю систему рівнянь і визначити вірогідність $P_{i,j}$ всіх станів моделі .

Навіть в разі простої моделі всього з 2-ма ФП розв'язати в загальному вигляді приведену систему рівнянь при довільних значеннях n скрутно і, навіть отримавши рішення, ними буде важко користуватися.

Розглянута вище модель конвеєрного суперскалярного ядра процесора з двома спеціалізованими функціональними пристроями є хорошим прикладом того, наскільки обмеженими є аналітичні методи при дослідженні складних комп'ютерних систем.

В реальних сучасних ядрах процесорів є не менше 6-8 спеціалізованих функціональних пристроїв.

При $m > 2$ ФП і довільному розмірі буферів заявок скрутно навіть представити граф станів моделі та систему рівнянь для вірогідності станів, а тим більше вирішити в загальному вигляді систему рівнянь.

Для дослідження таких моделей доцільно використовувати або імітаційне моделювання, або замінювати їх еквівалентними моделями, які дозволяють отримати аналітичні вирази з прийнятною точністю.

2.2.4 Модель ядра з 2 ФП в збалансованому режимі при $\rho_1 = 1,\ \rho_2 = 1$

Для деяких окремих випадків, наприклад, для ідеально збалансованих моделей з двома ФП і з $\rho_i = \frac{W_i \lambda}{\mu_i} = 1$ і однаковими буферами, рішення в загальному вигляді може бути отримане.

Для таких моделей вірогідності всіх станів при значних розмірах буферів будуть рівні

$$P_{ij} = \frac{1}{(n+2)^2 - 1}. \tag{2.29}$$

ПК блокований в усіх станах, коли один з індексів є $(n+1)$. Всього таких станів $2(n+1)$.

Вірогідність блокування (простою) ПК

$$P_{бл} = \sum_{i=0}^{n} P_{i,n+1} - \sum_{j=0}^{n} P_{m+1,j} = \frac{2(n+1)}{(n+2)^2 - 1}. \tag{2.30}$$

Коефіцієнт використання ПК

$$E_{ПК} = 1 - P_{бл} = 1 - \frac{2(n+1)}{(n+2)^2 - 1} = \frac{(n+1)^2}{(n+2)^2 - 1}. \tag{2.31}$$

Граничне значення коефіцієнта використання ПК при n $\to\infty$

$$\lim_{n\to\infty} E_{ПК} = \lim_{n\to\infty} \frac{(n+1)^2}{(n+2)^2} = 1. \tag{2.32}$$

ФП1 простоює в станах $a_{0,j}$ (j=0,1,2…..n+1), а ФП2 - простоює в станах $a_{i,0}$ (i=0,1,2…..n+1).

Вірогідність блокування (простою) ФП1

$$P_{бл} = \sum_{j=0}^{n+1} P_{0j} = \frac{n+2}{(n+2)^2 - 1}. \tag{2.33}$$

Коефіцієнт використання ФП1

$$H_{ФП1} = 1 - \sum_{j=0}^{n+1} P_{0j} = 1 - \frac{n+2}{(n+2)^2 - 1} = \frac{(n+2)(n+1)-1}{(n+2)^2 - 1}; \qquad (2.34)$$

Вірогідність блокування (простою) ФП2

$$P_{бл} = \sum_{j=0}^{n+1} P_{0j} = \frac{n+2}{(n+2)^2 - 1}. \qquad (2.35)$$

Коефіцієнт використання ФП2

$$H_{ФП2} = 1 - \sum_{i=0}^{n+1} P_{i0} = 1 - \frac{n+2}{(n+2)^2 - 1} = \frac{(n+2)(n+1)-1}{(n+2)^2 - 1}. \qquad (2.36)$$

Граничне значення коефіцієнта використання ФП при n $\to\infty$

$$\lim_{n\to\infty} H_{ФП} = \lim_{n\to\infty} \frac{(n+2)(n+1)-1}{(n+2)^2 - 1} = 1. \qquad (2.37)$$

Значення коефіцієнтів використання пристроїв як функції розміру буферів приведені в таблиці 2.4.

Таблиця 2.4 - Значення коефіцієнтів використання пристроїв як функції розміру буферів

n	0	2	4	8	16	32	64
E1	0.333	0.733	0.829	0.899	0.944	0.971	0.985
E2	0.333	0.733	0.829	0.899	0.944	0.971	0.985
H	0.333	0.500	0.714	0.818	0.895	0.943	0.970

В разі збалансованої системи з двома спеціалізованими функціональними пристроями при :

- при показових законах розподілу часів генерації та виконання команд:

- при збільшенні розміру буферів коефіцієнти використання пристроїв прагнуть до 1.

При буферах в 32 елементи досягаються коефіцієнти використання пристроїв 0.971.Подальше збільшення розміру буфера дуже мало збільшує коефіцієнти використання пристроїв.

- граничні значення коефіцієнтів використання ФП і ПК дорівнюють 1.
- при детермінованих часах генерації та виконання команд:
- буфери не потрібні;
- коефіцієнти використання пристроїв дорівнюють 1.

Отримання аналітичних результатів моделей ядра з багатьма спеціалізованими функціональними пристроями та довільними розмірами буферів в загальному вигляді шляхом складання та розвязання системи рівнянь практично неможливе.

2.2.5 Еквівалентна модель ядра з m ФП і індивідуальними буферами заявок для кожного ФП у вигляді m приватних моделей

Розглянемо еквівалентну модель ядра, який складається з ПК, m ФП з індивідуальними буферами заявок, у вигляді набору з m приватних моделей, які складаються з ПКі з інтенсивністю генерації команд i-го типу $W_i\lambda$; ФПі, - з інтенсивністю виконання команд i-го типу μ_i; індивідуального буфера заявок обсягом $n_i = W_i * n$ (рис. 2.4).

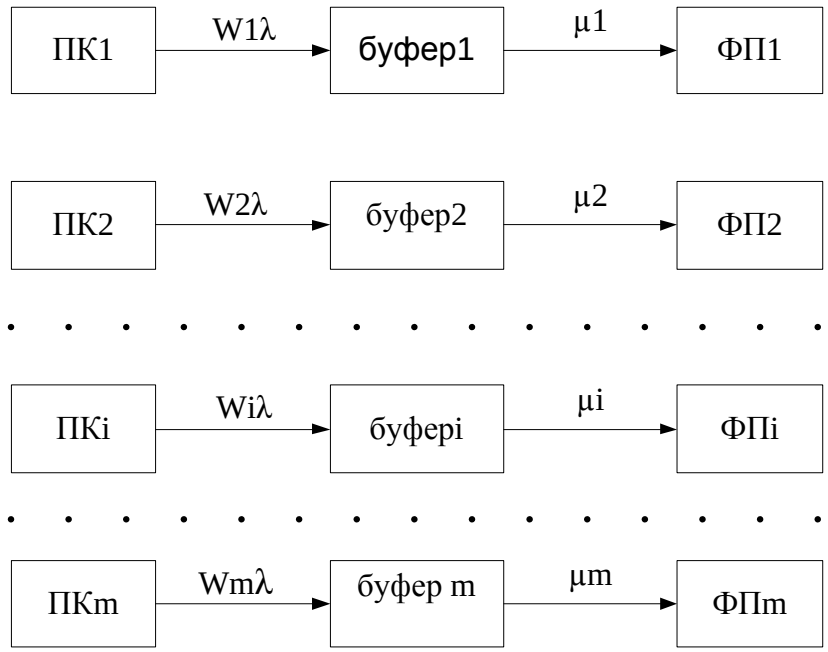


Рис. 2.4 - Структура моделі ядра з m ПК, m ФП з індивідуальними буферами заявок у вигляді набору з m приватних моделей

Модель з одним ПК, m ФП з індивідуальними буферами заявок (рис.4.1) і модель з m ПКі, m ФУі з індивідуальними буферами заявок у вигляді набору з m приватних моделей (рис. 2.4) будуть еквівалентними, якщо:

- продуктивності (інтенсивності генерації) підсистем генерації команд будуть однакові;

- продуктивності (інтенсивності виконання) підсистем виконання команд будуть однакові;

- алгоритми роботи підсистем генерації та виконання будуть ідентичними.

В обох моделях:

- сумарна інтенсивність генерації команд всіх типів однакова і є λ;

- інтенсивності виконання команд всіх типів також однакові.

Для забезпечення ідентичності алгоритмів генерації команд всіх типів в еквівалентній моделі з m приватних моделей (рис. 4.3) в разі блокування пристрою керуваня в одній з найбільш повільних приватних моделей, блокуються і всі інші пристрої керуваня швидших приватних моделей. Це збільшує час генерації заявок в них, що еквівалентно зменшенню інтенсивності генерації заявок λ_i.

Оскільки в еквівалентній моделі з m приватних моделей (рис. 4.4) виконано всіх 3 умови еквівалентності з моделлю із одним ПК (рис. 4.1), то, отже, можна чекати, що результати порівнюваних моделей будуть близькі.

Для оцінки показників ефективності кожної окремої підсистеми еквівалентної моделі (рис. 2.4) можна використовувати отримані раніше аналітичні вирази:

коефіцієнт використання функціонального пристрою:

$$H_i = \frac{\rho_i * (1 - \rho_i^{n+1})}{1 - \rho^{n+2}} \quad - \quad \text{при} \quad \rho_i \neq 1;$$

$$H_i = \frac{n_i + 1}{n_i + 2} \quad - \quad \text{при} \quad \rho_i = 1;$$

коефіцієнт звикористання пристрою керування:

$$E_i = \frac{1 - \rho_i^{n+1}}{1 - \rho_i^{n+2}} \quad - \quad \text{при} \quad \rho_i \neq 1;$$

$$E_i = \frac{n_i + 1}{n_i + 2} \quad - \quad \text{при} \quad \rho_i = 1;$$

$$\rho_i = \frac{\lambda_i}{\mu_i} = \frac{W_i * \lambda}{\mu_i}.$$

Граничні значення коефіцієнтів використання пристроїв при $n_i \to \infty$ будуть такі

$$\lim_{n\to\infty} H_i = H_i^{гран} = \rho_i; \qquad \lim_{n\to\infty} E_i = E_i^{гран} = 1 \text{ - при } \rho_i \le 1;$$

$$H_i^{гран} = 1; \qquad E_i^{гран} = \frac{1}{\rho_i} \text{ - при } \rho_i \succ 1.$$

В режимах з $\rho_i \le 1$ коефіцієнти використання функціональних пристроїв поводяться так же, як і в разі моделі з одним функціональним пристроєм (значення, близькі до граничних досягаються при розмірі буферів: $n \ge 32$ – при $\rho_i = 1$ та $n \ge 8$ - при $\rho_i \ne 1$, причому, чим значніша відмінність ρ_i від 1 в будь-яку сторону, тим менший розмір буфера потрібний. Похибки при використанні граничних значень коефіцієнтів використання замість реальних значень при буферах достатнього розміру (наприклад 32 та більше) не перевищують декількох відсотків.

Коефіцієнт використання пристрою керуваня в початковій моделі (рис. 2.1) буде

$$H_{ПК} = \sum_{i=1}^{m} W_i * H_i .$$

Граничне значення коефіцієнта використання пристрою керування в початковій моделі (рис. 2.1)

$$\lim_{n\to\infty} H_{ПК} = H_{ПК}^{пред} = \sum_{i=1}^{m} W_i * H_i^{гран} = \sum_{i=1}^{m} W_i * 1 = 1.$$

В режимах з хоч би одним функціональним пристроєм з $\rho_i > 1$ використовувати наведені вище формули для коефіцієнтів використання пристроїв не можна із-за великих похибок.

Хорошою моделлю є коригування інтенсивності генерації команд відповідно до формули

$$\lambda_{iкор} = \lambda_i * E_j \ ,$$

де:

- $\lambda_i = \dfrac{W_i * \lambda}{\mu_i}$;

- $E_j = 1 - P_{n+1}^j$ - коефіцієнт використання $ПК_j$ в найповільнішій приватній моделі (j);

- P_{n+1}^j - вірогідність блокування в найповільнішій приватній моделі j .

Найповільнішою приватною моделлю буде приватна модель з максимальним значенням коефіцієнта навантаження $\rho_j = \frac{\lambda_j}{\mu_j} = \frac{W_j * \lambda}{\mu_j}$ = max із $\rho_i = \frac{\lambda_i}{\mu_i} = \frac{W_i * \lambda}{\mu_i}$ - для всіх $i = 1,2,3....m, i \neq j$ (для визначеності нехай це буде приватна модель j).

Якщо $P_{n+1}^j \to 0$, а $H_j \to 1$, то блокування швидших приватних моделей з боку найповільнішої приватної моделі не буде.

2.2.6 Методика визначення коефіцієнтів використання функціональних пристроїв для всіх приватних моделей

Пропонується наступна методика визначення коефіцієнтів використання функціональних пристроїв для всіх приватних моделей.

Визначаються

$$\rho_i = \frac{\lambda_i}{\mu_i} = \frac{W_i * \lambda}{\mu_i} \quad \text{- для всіх } i = 1,2,3....m.$$

Серед всіх коефіцієнтів навантаження ρ_i визначається максимальний коефіцієнт навантаження (нехай це буде приватна модель j) більший 1

$$\rho_j = \frac{\lambda_j}{\mu_j} = \frac{W_j * \lambda}{\mu_j} \succ 1.$$

$ФП_j$ з максимальним коефіцієнтом навантаження $\rho_j \succ 1$ серед всіх функціональних пристроїв матиме і максимальний коефіцієнт використання H_j (близький до 1 при достатньому розмірі буфера). Коефіцієнти використання H_i функціональних пристроїв $ФП_i$ з $i = 1,2.....m, i \neq j$ ($\rho_i \prec \rho_j$) будуть меншими, ніж коефіцієнт використання H_j $ФП_j$, оскільки вони частину часу в порівнянні з $ФП_j$ простоюватимуть.

Для j приватної моделі визначаються коефіцієнти використання пристроїв по формулах

$$H_j = \frac{\rho_j * (1 - \rho_j^{n+1})}{1 - \rho_j^{n+2}}; \qquad E_j = \frac{(1 - \rho_j^{n+1})}{1 - \rho_j^{n+2}} .$$

Для всіх приватних моделей з номерами $i = 1,2.....m, i \neq j$:

- коригуються інтенсивності генерації заявок і коефіцієнти навантаження

$$\lambda_{ікор} = \lambda_i * E_j = (W_i * \lambda) * E_j ;$$

$$\rho_{ікор} = \frac{\lambda_{ікор}}{\mu_i} = \frac{W_i * \lambda}{\mu_i} * E_j = \rho_i * E_j ;$$

- визначаються скореговані значення коефіцієнтів використання функціональних пристроїв

$$H_{ікор} = \frac{\rho_{ікор} * (1 - \rho_{ікор}^{n+1})}{1 - \rho_{ікор}^{n+2}} ;$$

$$E_{ікор} = \frac{1 - \rho_{ікор}^{n+1}}{1 - \rho_{ікор}^{n+2}} .$$

Коефіцієнт використання пристрою керуваня в початковій моделі з одним ПК і багатьма ФП

$$E_{ПК} = E_j .$$

При виборі досить великих буферів значення коефіцієнтів використання $ФП_i$ з малими похибками можна замінити їх граничними значеннями

$$\lim_{n\to\infty} E_{ПК} = E_j^{гран} = \frac{1}{\rho_j}; \quad \lim_{n\to\infty} H_j = H_j^{гран} = 1 \text{ - оскільки } \rho_j \succ 1;$$

$$\lim_{n\to\infty} H_{ікор} = H_{ікор}^{пред} = \rho_{ікор} = \rho_i * E_j = \frac{\rho_i}{\rho_j}; \quad \lim_{n\to\infty} E_{ікор} = E_{ікор}^{гран} = 1 \text{ - при } \rho_{ікор} \prec 1.$$

2.2.7 Перевірка правомірності використання спрощеної моделі для дослідження ефективності систем з одним ПК і довільною кількістю ФП

В таблицях 2.5 - 2.13 приведені результати дослідження моделей з одним пристроєм керуваня і 2-, 3 - та 4-мя функціональними пристроями.

В таблицях використані такі скорочення:

- $Е_{ПК}$ - коефіцієнт використання пристрою керуваня.

- $Н_{ФПі}$ - коефіцієнт використання функціонального пристрою з номером i.

- Еталонна модель - результати імітаційного моделювання.

- Спрощена модель - результат розрахунків відповідно до приведеної вище методики визначення коефіцієнтів використання пристроїв в системі приватних моделей.

- n - розмір буфера заявок**;**

- ρ_i - коефіцієнт навантаження $ФП_i$.

Таблиця 2.5 - Результати моделей з 2 ФП при $\rho_1 = 0.5; \rho_2 = 0.6$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,731	0,851	0,364	0,365	0,433	0,438
2	0,853	0,914	0,429	0,427	0,506	0,512
4	0,954	0,969	0,481	0,477	0,568	0,572
8	0,996	0,996	0,502	0,498	0,596	0,597
16	1,000	1,000	0,505	0,500	0,597	0,600
32	1,000	1,000	0,505	0,500	0,597	0,600

Таблиця 2.6 - Результати моделей з 2 ФП при $\rho_1 = 1.2; \rho_2 = 0.8$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,516	0,706	0,628	0,625	0,409	0,484
2	0,62	0,753	0,743	0,747	0,500	0,542
4	0,726	0,794	0,863	0,868	0,579	0,600
8	0,793	0,820	0,952	0,954	0,642	0,641
16	0,833	0,830	0,993	0,992	0,665	0,661
32	0,825	0,833	1,000	1,000	0,658	0,666

Таблиця 2.7 - Результати моделей з 2 ФП при $\rho_1 = 1.5; \rho_2 = 0.6$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,492	0,597	0,727	0,731	0,291	0,316
2	0,562	0,629	0,855	0,849	0,335	0,351
4	0,627	0,652	0,946	0,946	0,377	0,381
8	0,660	0,663	0,992	0,991	0,392	0,396
16	0,667	0,666	1,000	1,000	0,400	0,400
32	0.667	0.666	1,000	1,000	0,400	0,400

Таблиця 2.8 - Результати моделей при $\rho_1 = 2$; $\rho_2 = 1.5$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,339	0,429	0,681	0,684	0,506	0,643
2	0,401	0,467	0,798	0,805	0,597	0,700
4	0,456	0,492	0,913	0,915	0,678	0,738
8	0,489	0,500	0,976	0,979	0,734	0,749
16	0,500	0,500	0,997	0,998	0,749	0,750
32	0,500	0,500	1,000	1,000	0,750	0,750

Таблиця 2.9 - Результати моделей з 3 ФП при $\rho_2 = 1.5$; $\rho_2 = 0.6$; $\rho_3 = 0.8$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$		$H_{П3}$	
	Еталонна модель	Спро-щена	Еталонна модель	Спро-щена	Еталонна модель	Спро-щена	Еталонна модель	Спро-щена
1	0,437	0,526	0,659	0,700	0,260	0,316	0,348	0,421
2	0,534	0,585	0,802	0,824	0,318	0,351	0,423	0,468
4	0,621	0,635	0,924	0,934	0,373	0,381	0,502	0,508
8	0,660	0,661	0,989	0,989	0,397	0,396	0,537	0,529
16	0,673	0,666	1,000	1,000	0,407	0,400	0,533	0,533
32	0,666	0,667	1,000	1,000	0,404	0,400	0,528	0,533

Таблиця 2.10 - Результати моделей з 3 ФП при $\rho_1 = 2$; $\rho 2 = 1.5$; $\rho_3 = 0.8$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$		$H_{П3}$	
	Еталонна модель	Спро-щена модель	Еталонна модель	Спро-щена модель	Еталонна модель	Спро-щена модель	Еталонна модель	Спро-щена модель
1	0,319	0,429	0,647	0,684	0,474	0,643	0,258	0,343
2	0,389	0,467	0,772	0,805	0,592	0,700	0,312	0,373
4	0,453	0,492	0,903	0,915	0,677	0,738	0,359	0,394
8	0,487	0,500	0,976	0,979	0,733	0,749	0,39	0,400
16	0,498	0,500	0,997	0,998	0,747	0,750	0,395	0,400
32	0,501	0,500	1,000	1,000	0,740	0,750	0,397	0,400

Таблиця 2.11 - Результати моделей з 3 ФП при $\rho_1 = 2$; $\rho 2 = 1.5$; $\rho_3 = 1.8$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$		$H_{П3}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,274	0,429	0,545	0,641	0,412	0,643	0,497	0,771
2	0,340	0,467	0,681	0,756	0,518	0,700	0,614	0,840
4	0,407	0,492	0,830	0,865	0,611	0,738	0,742	0,886
8	0,462	0,500	0,928	0,940	0,685	0,749	0,827	0,899
16	0,485	0,500	0,974	0,980	0,722	0,750	0,887	0,900
32	0,496	0,500	0,998	0,997	0,745	0,750	0,896	0,900

Таблиця 2.12 - Результати моделей з 4 ФП при $\rho_1 = 2$; $\rho 2 = 1.5$; $\rho_3 = 0.5$; $\rho_4 = 0.5$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$		$H_{П3}$		$H_{ФП4}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,320	0,429	0,644	0,684	0,480	0,643	0,157	0,214	0,157	0,214
2	0,396	0,467	0,788	0,805	0,600	0,700	0,196	0,233	0,196	0,233
4	0,455	0,492	0,914	0,915	0,670	0,738	0,227	0,246	0,227	0,246
8	0,479	0,500	0,979	0,979	0,720	0,749	0,238	0,250	0,238	0,250
16	0,500	0,500	0,997	0,998	0,760	0,750	0,244	0,250	0,244	0,250
32	0,502	0,500	1,000	1,000	0,740	0,750	0,251	0,250	0,251	0,250

Таблиця 2.13 - Результати моделей з 4 ФП при $\rho_1 = 2; \rho_2 = 1.5; \rho_3 = 1.8; \rho_4 = 1.8$

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$		$H_{П3}$		$H_{ФП4}$	
	Ета-лонна модель	Спро-щена модель	Ета-лонна модель	Спро-щена модель	Ета-лонна модель	Спро-щена модель	Ета-лонна модель	Спро-щена модель	Ета-лонна модель	Спро-щена модель
1	0,236	0,429	0,476	0,641	0,352	0,64	0,420	0,770	0,420	0,770
2	0,304	0,467	0,605	0,756	0,453	0,700	0,550	0,840	0,550	0,840
4	0,381	0,492	0,759	0,865	0,578	0,740	0,690	0,890	0,690	0,890
8	0,443	0,500	0,888	0,940	0,655	0,750	0,800	0,900	0,800	0,900
16	0,473	0,500	0,951	0,980	0,707	0,750	0,860	0,900	0,860	0,900
32	0,493	0,500	0,995	0,997	0,727	0,750	0,870	0,900	0,870	0,900

З аналізу таблиць 2.5 - 2.13 витікає, що при виборі буфера $n \geq 32$ похибки результатів спрощеної моделі в порівнянні з результатами імітаційного моделювання складають не більше 5%.

Отже, можна зробити висновок про те, що методика визначення коефіцієнтів використання пристроїв через дослідження єквивалентгої моделі, яка складається з m приватних моделей, добре працює як для 2, 3, 4 ФП, так і для довільної кількості функціональних пристроїв в широкому діапазоні зміни параметрів моделі.

2.7 Висновки

При показових законах генерації і виконання команд:

1. Коефіцієнт використання ФП:

$$H_{ФП} = \frac{\rho(1-\rho^{n+1})}{1-\rho^{n+2}} \quad \text{- при} \quad \rho \neq 1, \tag{2.19}$$

$$H_{ФП} = \frac{n+1}{n+2} \quad \text{- при} \quad \rho = 1. \tag{2.20}$$

2. Коефіцієнт використання ПК:

$$E_{ПК} = \frac{1-\rho^{n+1}}{1-\rho^{n+2}} \quad \text{- при} \quad \rho \neq 1, \tag{2.21}$$

$$E_{ПК} = \frac{n+1}{n+2} \quad - \quad \text{при} \quad \rho = 1. \qquad (2.22)$$

3. Для збалансованої моделі при $\rho = 1$ граничні значення коефіцієнтів використання ФП і ПК дорівнюють 1. При буфері в 32 елементи досягаються коефіцієнти використання пристроїв рівні 0,971. Подальше збільшення розміру буфера незначно збільшує коефіцієнти використання пристроїв.

4. В не збалансованому режимі при $\rho > 1$:

1) коефіцієнт використання менш продуктивного пристрою моделі (а ним є ФП) швидко прагне до свого граничного значення рівного 1 (тим швидше, чим більш система незбалансована). Так, для $\rho = 2$ при $n = 4$ коефіцієнт використання ФП вже практично рівний 1, а при $\rho = 1.1$ близький до граничного значення при $n = 8$;

2) коефіцієнт використання ПК (продуктивнішого елементу моделі) швидко прагне до свого граничного значення, рівного $1/\rho$. Залежність наближення до граничного значення така ж як і для коефіцієнта використання ФП.

Таким чином, буфер $n \geq 8$ в системі забезпечить коефіцієнти використання пристроїв, близькі до граничних значень.

5. В не збалансованому режимі при $\rho < 1$ коефіцієнт використання менш продуктивного пристрою моделі (а ним є ПК) швидко прагне до свого граничного значення, рівного 1 (тим швидше, чим більш система незбалансована). Так, для $\rho = 0.5$ при $n = 4$ коефіцієнт використання ПК вже практично дорівнює 1, а при $\rho = 0.9$ близький до граничного значення при $n = 8$.

Поведінка коефіцієнта використання ФП (продуктивнішого елементу моделі) залежно від розміру буфера при різних (аналогічна поведінці коефіцієнта використання ПК, тільки граничне значення буде ρ.

Таким чином, і в цьому випадку буфер $n \geq 8$ в системі забезпечить коефіцієнти використання пристроїв, близькі до граничних значень.

6. Буфер $n \geq 32$ забезпечує коефіцієнти використання пристроїв близькі до граничних при будь-яких значеннях ρ.

7. При детермінованих часах генерації і виконання команд:

1) Коефіцієнт використання ФП:

$$H_{ФП} = \rho \quad - \quad \text{при} \quad \rho < 1, \qquad (2.23)$$

$$H_{ФП} = 1 \qquad \text{- при} \quad \rho \geq 1. \tag{2.24}$$

2) Коефіцієнт використання ПК:

$$E_{ФП} = \frac{1}{\rho} \qquad \text{- при} \quad \rho > 1, \tag{2.25}$$

$$E_{ФП} = 1 \qquad \text{- при} \quad \rho \leq 1. \tag{2.26}$$

8. Закони зміни коефіцієнтів використання пристроїв при показових законах генерації і виконання команд аналогічні законам зміни коефіцієнтів використання пристроїв при детермінованих часах генерації і виконання команд.

9. Граничні значення коефіцієнтів використання пристроїв при показових законах генерації і виконання команд співпадають зі значеннями коефіцієнтів використання пристроїв при детермінованих часах генерації і виконання команд.

10. Похибка показників ефективності при порівнянні аналітичних результатів з результатами імітаційного моделювання складає не більше 1% .

Для визначення коефіцієнтів використання пристрою керуваня та m спеціалізованих функціональних пристроїв можна використовувати аналітичні результати, отримані при дослідженні набору приватних моделей з коригуванням їх, якщо це необхідно.

Для цього:

- початкова модель з m функціональними пристроями замінюється на m приватних моделей. Пристрій керуваня в кожній приватній моделі генерує команди для функціонального пристрою з інтенсивністю $\lambda * W_i$, де:

W_i - це частота появи команд i-го типу;

λ - інтенсивність генерації команд всіх типів;

μ_i - інтенсивність виконання команд $ФП_i$;

$\rho_i = \dfrac{\lambda * W_i}{\mu_i}$ - коэффицент навантаження $ФП_i$.

- якщо коефіцієнти навантаження $ФП_i$ всіх приватних моделей менші або рівні 1 ($\rho_i \leq 1$ для всіх $i = 1, 2.....m$), то коефіцієнти використання функціональних пристроїв $ФП_i$ визначаються по формулах;

$$H_i = \frac{\rho_i * (1-\rho_i{}^{n+1})}{1-\rho^{n+2}} \quad - \quad \text{при} \quad \rho_i \prec 1, \tag{2.27}$$

$$H_i = \frac{n_i + 1}{n_i + 2} \quad - \quad \text{при} \quad \rho_i = 1. \tag{2.28}$$

- коефіцієнти використання $ПК_i$ визначаються по формулах;

$$E_i = \frac{1-\rho_i{}^{n+1}}{1-\rho_i{}^{n+2}} \quad - \quad \text{при} \quad \rho_i \neq 1, \tag{2.29}$$

$$E_i = \frac{n_i + 1}{n_i + 2} \quad - \quad \text{при} \quad \rho_i = 1. \tag{2.30}$$

- коефіцієнт використання $ПК$ пристрою керуваня в початковій моделі з одним ПК і багатьма ФП визначається по формулі

$$E_{ПК} = \sum_{i=1}^{m} W_i * E_{ПКi}. \tag{2.31}$$

При виборі досить великих буферів значення коефіцієнтів використання пристроїв з дуже маленькими погрішностями можна замінити їх граничними значеннями

$$\lim_{n\to\infty} E_{ПКi} = 1; \quad \lim_{n\to\infty} H_{ФПi} = \rho_i \text{ - тому що } \rho_i \prec 1; \quad \lim_{n\to\infty} E_{ПК} = 1. \tag{2.32}$$

Якщо серед коефіцієнтів навантаження $ФП_i$ приватних моделей є коефіцієнти $\rho_i \succ 1$ (не має значення всі коефіцієнти або частина коефіцієнтів), то:

- серед всіх коефіцієнтів навантаження ρ_i знаходиться максимальний коефіцієнт навантаження (для визначеності нехай це буде приватна модель j) більший 1

$$\rho_j = \frac{\lambda_j}{\mu_j} = \frac{W_j * \lambda}{\mu_j} \succ 1. \tag{2.33}$$

- для приватної моделі j визначаються коефіцієнти використання пристроїв по формулах

$$H_{ФПj} = \frac{\rho_j * (1-\rho_j^{n+1})}{1-\rho_j^{n+2}}, \tag{2.34}$$

$$E_{ПКj} = \frac{(1-\rho_j^{n+1})}{1-\rho_j^{n+2}}. \tag{2.35}$$

- для всіх приватних моделей з номерами $i = 1, 2.....m, i \neq j$:

- коригуються інтенсивності генерації заявок і коефіцієнти навантаження

$$\lambda_{ікор} = \lambda_i * E_{ПКj} = (W_i * \lambda) * E_{ПКj}\,, \tag{2.36}$$

$$\rho_{ікор} = \frac{\lambda_{ікор}}{\mu_i} = \frac{W_i * \lambda}{\mu_i} * E_{ПКj} = \rho_i * E_{ПКj}\,. \tag{2.37}$$

- визначаються скореговані значення коефіцієнтів використання функціональних пристроїв $ФП_i$

$$H_{ФПікор} = \frac{\rho_{ікор} * (1 - \rho_{ікор}^{n+1})}{1 - \rho_{ікор}^{n+2}}\,, \tag{2.38}$$

$$E_{ПКікор} = \frac{1 - \rho_{ікор}^{n+1}}{1 - \rho_{ікор}^{n+2}}\,. \tag{2.39}$$

- коефіцієнт використання $E_{ПК}$ пристрою керуваня в початковій моделі з одним ПК і багатьма ФП визначається по формулі

$$E_{ПК} = E_{ПКj}\,. \tag{2.40}$$

При виборі досить великих буферів значення коефіцієнтів використання пристроїв з незначними похибками можна замінити їх граничними значеннями

$$\lim_{n\to\infty} E_{ПК} = E_{ПКj}^{пред} = \frac{1}{\rho_j}\,, \tag{2.41}$$

$$\lim_{n\to\infty} H_{ФПj} = H_{ФПj}^{пред} = 1 \quad \text{- оскільки} \quad \rho_j \succ 1\,, \tag{2.42}$$

$$\lim_{n\to\infty} H_{ФПікор} = H_{ФПікор}^{пред} = \rho_i * E_{ПКj} = \frac{\rho_i}{\rho_j}\,. \tag{2.43}$$

РОЗДІЛ 3
ДОСЛІДЖЕННЯ ВПЛИВУ КОМАНД ПЕРЕХОДІВ НА ПРОДУКТИВНІСТЬ КОНВЕЄРНИХ ЯДЕР ПРОЦЕСОРІВ

3.1 Залежності по керуванню

Для підвищення продуктивності ядер сучасних процесорів широко використовується конвеєрний принцип. Велика довжина конвеєра сприяє збільшенню тактової частоти, і, як наслідок, створює потенційну можливість зростання продуктивності. В ідеальному конвеєрі час роботи усіх етапів однаковий і відсутні "перешкоди" роботі конвеєра. В реальних конвеєрах не вдається забезпечити однакові часи роботи всіх етапів, а також уникнути впливу "перешкод", таких як структурні конфлікти, залежності за даними, залежністю по керуванню.

Структурні конфлікти вирішуються досить просто - розмножуються спільні ресурси.

Залежності за даними, якщо це можливо, усуваються за допомогою технологій перейменування регістрів, позачергового виконання команд.

Залежності по керуванню виникають при виявленні команд переходу, як безумовних, так і умовних.

При виконанні будь-якої команди переходу виконується перезавантаження конвеєра підготовки команд для виконання. При виконанні команди умовного переходу необхідно мати ознаку переходу, і тільки після її аналізу перезавантажувати конвеєр, якщо це необхідно.

Очікування формування ознаки переходу (причому, чим більше етапів у конвеєра, тим більший час очікування) і перезавантаження конвеєра підготовки команд призводять до простоїв дешифратора команд і підсистемивиконання, а отже, і до зниження продуктивності всієї системи.

Втрати часу на перезавантаження конвеєра підготовки команд для виконання можуть бути мінімізовані за рахунок реалізації швидкодіючих підсистем кеш-пам'яті, спеціальних буферів команд.

Для вибору оптимального (чи близького до нього) варіанту структури ядра необхідно знати залежності ефективності ядра від його різних параметрів (в тому числі і від частоти появи команд переходів в программи), бажано у вигляді аналітичних виразів.

Метою дослідження в цьому розділі є отримання аналітичних оцінок впливу команд переходів на ефективність роботи конвеєра підготовки команд.

3.2 Функціонування ядра при наявності команд переходів

При виявленні команди умовного переходу конвеєр підготовки команд блокується до появи ознаки переходу або результату операції. Після виконання підсистемою обробки (функціональним пристроєм) всіх раніше підготовлених команд є ознака переходу, в результаті аналізу якої формується адреса першої команди нової гілки програми і читається з підсистеми пам'яті перша команда нової гілки. Після отримання цієї команди з підсистеми пам'яті вона дешифрується і заноситься в буфер. Час перезавантаження конвеєра підготовки команд (час простою дешифратора команд) при умовному переході до обробки першої команди нової гілки програми складе

$$T_{пер} = t_{оч} + t_{фа} + t_{чт.пам} + t_{зап.кон} . \qquad (3.1)$$

де:

- $t_{оч}$ - час очікування формування ознаки переходу (час завершення раніше згенерованих команд при виконанні команд по порядку);

- $t_{фа}$ - час формування адреси першої команди нової гілки програми;

- $t_{чт.пам}$ - час читання першої команди нової гілки програми;

- $t_{зап.кон}$ - час заповнення конвеєра команд до дешифратора.

3.3. Вибір методу дослідження

Дослідження впливу команд переходів проводиться з використанням аналітичного методу з наступною перевіркою точності за допомогою імітаційного моделювання та уточнення отриманих результатів.

3.4. Модель ядра процесора

Як відомо, будь-яка модель об'єкту - це спрощене представлення основних особливостей цього об'єкту. Модель має бути максимально простою, щоб забезпечити її дослідження аналітичними методами, і при цьому відображувати основні, істотні для дослідження риси об'єкту.

В розробленій моделі збережена така особливість структури ядра як конвейерність. Для цього в один етап конвеєра об'єднані всі етапи підготовки команд включаючи дешифратор команд. Інший етап об'єднує всі етапи підсистеми виконання команд. Для цілей дослідження байдуже чи є підсистема виконання (функціональний пристрій) команд конвеєрною або комбінаційною. За відсутності блоку передбачення або при "невдалому" передбачені напряму переходу для набуття ознаки переходу необхідно дочекатися виконання всіх команд, які були згенеровані перед командою умовного переходу. При цьому не має значення чи виконувалися ці команди строго по порядку в підсистемі виконання, чи було використано позачергове виконання команд в спеціалізованих функціональних пристроях, оскільки результати виконання команд записуються в архітектурні регістри строго в порядку передбаченому програмою, тобто послідовно в порядку номерів команд програми. Структура ядра із спеціалізованими функціональними пристроями може бути замінена універсальним функціональним пристроєм з еквівалентною продуктивністю.

Структура моделі ядра з урахуванням допущень представлена на рис. 3.1.

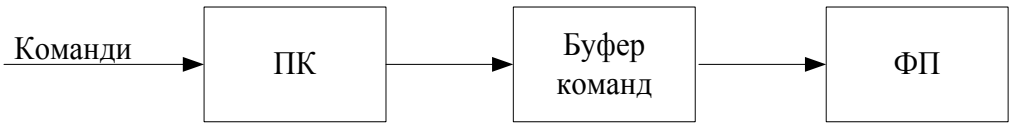


Рис. 3.1 - Структура моделі ядра процесора

де ПК – пристрій керування; ФП – універсальний функціональний пристрій

Зробимо такі припущення:

- на вхід першої стадії (етапу) поступає послідовність команд програми таких типів:

- команди обробки та команди звернення в підсистему пам'яті - з вірогідністю ω_O;

- команди умовного переходу - з вірогідністю ω_n $(\omega_n + \omega_0 = 1)$;

- команди обробки, команди звернення в підсистему пам'яті виконуються послідовно в в підсистемі виконання, представленій як універсальний функціональний пристрій – ФП (друга стадія моделі);

- перша стадія моделі ПК генерує потік команд (заявок) з інтенсивністю λ;

- між стадіями є буфер на n заявок.

- потік заявок, згенерований ПК найпростіший з експоненціальним законом розподілу. Якщо буфер заявок заповнений, то ПК блокується, тобто зупиняється.

Підсистема виконання (ФП) виконує заявки (команди) з інтенсивністю μ. Закон розподілу часу виконання заявок експоненціальний. Дисципліна вибору заявок з буфера - FIFO. За відсутності підготовлених до виконання команд (заявок) ФП простоює. Якщо буфер порожній і ПК згенерував чергову заявку, то вона миттєво поступає у ФП.

При виявленні чергової команди переходу відбувається перезавантаження конвеєра підготовки команд з інтенсивністю μ_n а при перезавантаженні конвеєра підготовки команд ПК блокується (простоює). Підсистема виконання (ФП) за наявності готових до виконання команд виконує їх (працює).

3.5 Стани моделі та система рівнянь

Граф станів моделі при блокуванні конвеєра відразу при виявленні «перешкоди» приведений на рис. 3.2.

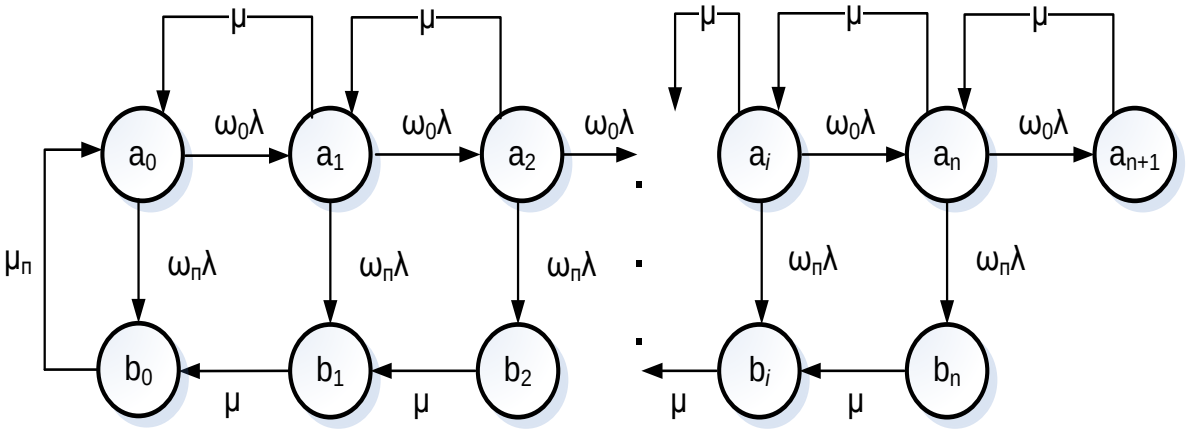


Рис. 3.2 - Граф станів моделі при блокуванні конвеєра відразу при виявленні «перешкоди»

ω_o - частота появи команд обробки;

μ - інтенсивність виконання команд обробки в функціональному пристрої;

λ - інтенсивність генерації команд ПК в режимі заповненого конвеєра (сталий режим), не залежить від типу команд.

Стани моделі будуть ідентифікуватися за допомогою індекса i $(0 \le i \le n+1)$, який відображає кількість згенерованих першою стадією (ПК) команд (заявок). Заявки можуть знаходитися - одна на виконанні в ФП, $(i-1)$ - в буфері. ПК і ФП можуть або працювати, або простоювати.

В станах:

- a_i $(1 \le i \le n)$ – ПК і ФП працюють; в буфері (i-1) заявка;

- a_0 - ПК працює, а ФП простоює через відсутність заявок; буфер пустий;

- a_{n+1} - ПК простоює, оскільки буфер заповнений, ФП працює;

- b_i $(1 \le i \le n)$ – ПК блокований, оскільки він чекає завершення всіх раніше згенерованих заявок, ФП працює; в буфері (i-1) заявка»

- b_0 - виконується перезавантаження конвеєра підготовки команд для дешифратора , ПК і ФП не працюють, буфер пустий.

В кожному з перерахованих станів система може знаходитися з деякою вірогідністю. Рівняння балансу вірогідності для станів такі:

$$P_{a0}\lambda = P_{b0}\mu + P_{a1}\mu\,, \tag{3.2}$$

$$P_{ai}(\lambda+\mu) = P_{ai-1}\omega_0\lambda + P_{a(i+1)}\mu \qquad (1 \le i \le n)\,, \tag{3.3}$$

$$P_{an+1}\mu = P_{an}\omega_0\lambda\,, \tag{3.4}$$

$$P_{b0}\mu = P_{a0}\omega_n\lambda + P_{b1}\mu\,, \tag{3.5}$$

$$P_{bi}\mu = P_{ai}\omega_n\lambda + P_{b(i+1)}\mu \qquad (1 \le i \le n-1)\,, \tag{3.6}$$

$$P_{bn}\mu = P_{an}(1-q)\omega_n\lambda\,. \tag{3.7}$$

Як показники ефективності використовуються коефіцієнти використання ПК та ФП.

Коефіцієнт використання ПК дорівнює сумі вірогідності всіх станів, в яких він не блокований.

Коефіцієнт використання ФП дорівнює сумі вірогідності всіх станів, в яких ФП не простоює.

3.6. Аналітичне рішення системи рівнянь

З одноманітного визначення вірогідності станів **ai** (i = 1, n+1) та bi (i = 1, n) виходить, що:

$$P_{ai} = R^i \cdot P_{a0}, \tag{3.8}$$

$$P_{bi} = R^{i-1} \cdot P_{b1} \qquad (1 \le i \le n), \tag{3.9}$$

Коефіцієнт навантаження R за наявності команд переходу визначається з рівняння

$$P_{ai} \cdot (\lambda + \mu) = P_{a(i-1)} \cdot \omega_0 \cdot \lambda + P_{a(i+1)} \cdot \mu . \tag{3.10}$$

Після ділення на μ отримаємо

$$P_{ai} \cdot (1 + \rho) = P_{a(i-1)} \cdot \omega_0 \cdot \rho + P_{a(i+1)} . \tag{3.11}$$

$\rho = \frac{\lambda}{\mu}$ - коефіцієнт навантаження на функціональний пристрій за відсутності команд переходів.

Після підстановки

$$P_{ai} = R \cdot P_{a(i-1)} . \tag{3.12}$$

отримаємо

$$R \cdot P_{a(i-1)} \cdot (1 + \rho) = P_{a(i-1)} \cdot \omega_0 \cdot \rho + R^2 \cdot P_{a(i-1)}, \tag{3.13}$$

$$P_{a(i-1)} \cdot (R^2 - R \cdot (1 + \rho) + \omega_0 \cdot \rho) = 0, \tag{3.14}$$

$$P_{a(i-1)} \neq 0. \tag{3.15}$$

Звідси

$$R^2 - R \cdot (1 + \rho) + \omega_0 \cdot \rho = 0 . \tag{3.16}$$

Корені квадратного рівняння будуть

$$R = \frac{(1+\rho) \pm \sqrt{(1+\rho)^2 - 4 \cdot \omega_0 \cdot \rho}}{2} \ . \tag{3.17}$$

Умовам завдання задовольняє корінь

$$R = \frac{(1+\rho) - \sqrt{(1+\rho)^2 - 4 \cdot \omega_0 \cdot \rho}}{2}. \tag{3.18}$$

Вірогідність P_{b0} визначається з рівняння

$$P_{a0} \cdot \lambda = P_{b0} \cdot \mu_n + \mu \cdot P_{a1}. \tag{3.19}$$

Розділивши обидві частини рівняння на μ і підставивши μ отримаємо

$$P_{a0} \cdot \rho = P_{b0} \cdot \frac{\mu_n}{\mu} + R \cdot P_{a0}, \tag{3.20}$$

$$P_{a0} \cdot (\rho - R) = P_{b0} \cdot \frac{\mu_n}{\mu}, \tag{3.21}$$

$$P_{b0} = \frac{(\rho - R)}{A} \cdot P_{a0} \tag{3.22}$$

, де

$$A = \frac{\mu_n}{\mu} * \frac{\lambda}{\lambda} = \frac{\lambda}{\mu} * \frac{\mu_{пер}}{\lambda} = \frac{t_{вик}}{t_{ген}} * \frac{t_{ген}}{t_{пер}} = \rho * \rho_n, \tag{3.23}$$

$$\rho = \frac{\lambda}{\mu}, \ \rho_n = \frac{\mu_n}{\lambda}. \tag{3.24}$$

P_{b1} визначається з рівняння

$$P_{b0} \cdot \mu_n = P_{a0} \cdot \omega_n \cdot \lambda + P_{b1} \cdot \mu, \tag{3.24}$$

$$P_{b0} \cdot \frac{\mu_n}{\mu} = P_{a0} \cdot \omega_n \cdot \rho + P_{b1}, \tag{3.25}$$

$$P_{b1} = A \cdot P_{b0} - \omega_n \cdot \rho \cdot P_{a0} = A \cdot \frac{(\rho - R)}{A} \cdot P_{a0} - \omega_n \cdot \rho \cdot P_{a0}, \tag{3.26}$$

$$P_{b1} = \left[(\rho - R) - \omega_n \cdot \rho\right] \cdot P_{a0} = \left[\rho \cdot (1 - \omega_n) - R\right] \cdot P_{a0}, \tag{3.27}$$

$$P_{b1} = (\omega_0 \cdot \rho - R) \cdot P_{a0}, \tag{3.28}$$

$$P_{b0} = \frac{(\rho - R)}{A} \cdot P_{a0}, \tag{3.29}$$

$$P_{bi} = R^{i-1} \cdot P_{b1},\ (2 \le i \le n), \tag{3.30}$$

$$\sum_{i=0}^{n+1} P_{ai} + \sum_{i=0}^{n} P_{bi} = 1 \tag{3.31}$$

$$\sum_{i=0}^{n+1} P_{ai} = \frac{(1-R^{n+2})}{(1-R)} \cdot P_{a0} \tag{3.32}$$

$$\begin{aligned} &\sum_{i=1}^{n} P_{bi} = \sum_{i=1}^{n} R^{i-1} \cdot P_{b1} = \sum_{i=1}^{n} R^{i-1} \cdot (\omega_0 \cdot \rho - R) \cdot P_{a0} = \\ &= (\omega_0 \cdot \rho - R) \cdot \frac{(1-R^n)}{(1-R)} \cdot P_{a0}, \end{aligned} \tag{3.33}$$

$$1 = \left[\frac{(1-R^{n+2})}{(1-R)} + \frac{(\rho - R)}{A} + \frac{(\omega_0 \cdot \rho - R)}{(1-R)} \cdot (1-R^n) \right] \cdot P_{a0}. \tag{3.34}$$

Після підстановки виразу для P_{a0} отримаємо

$$P_{a0} = \frac{(1-R)}{(1-R^{n+2}) + \frac{(\rho - R) \cdot (1-R)}{A} + (\omega_0 \cdot \rho - R) \cdot \left(1-R^n\right)}, \tag{3.35}$$

$$E_{ПК} = \sum_{i=0}^{n} P_{ai} = \sum_{i=0}^{n} R^i \cdot P_{a0} = \frac{(1-R^{n+1})}{(1-R)} \cdot P_{a0} \tag{3.36}$$

Граничне значення коефіцієнта використання пристрою керування при $n \to \infty$ буде

$$\lim_{n \to \infty} E_{ПК} = \frac{1}{1 + \frac{(\rho - R) \cdot (1-R)}{A} + (\omega_0 \cdot \rho - R)}. \tag{3.37}$$

Коефіцієнт використання функціонального пристрою

$$H_{ФП} = 1 - (P_{a0} + P_{b0}), \tag{3.38}$$

$$\begin{aligned} &P_{a0} + P_{b0} = P_{a0} \cdot \left(1 + \frac{\rho - R}{A}\right) = \\ &= \frac{(1-R) \cdot \left(1 + \frac{\rho - R}{A}\right)}{(1-R^{n+2}) + \frac{(\rho - R) \cdot (1-R)}{A} + (\omega_0 \cdot \rho - R) \cdot \left(1-R^n\right)}, \end{aligned} \tag{3.39}$$

$$H_{ФП} = 1 - \frac{(1-R)\cdot\left(1+\frac{\rho-R}{A}\right)}{(1-R^{n+2}) + \frac{(\rho-R)\cdot(1-R)}{A} + (\omega_0\cdot\rho-R)\cdot\left(1-R^n\right)}, \quad (3.40)$$

$$H_{ФП} = \frac{1+R^{n+2} + \frac{(\rho-R)\cdot(1-R)}{A} + (\omega_0\cdot\rho-R)\cdot(1-R^n) - 1 + R - \frac{(\rho-R)\cdot(1-R)}{A}}{(1-R^{n+2}) + \frac{(\rho-R)\cdot(1-R)}{A} + (\omega_0\cdot\rho-R)\cdot\left(1-R^n\right)} \quad (3.41)$$

Після приведення подібних членів отримаємо

$$H_{ФП} = \frac{R\cdot(1-R^{n+1}) + (\omega_0\cdot\rho-R)\cdot(1-R^n)}{(1-R^{n+2}) + \frac{(\rho-R)\cdot(1-R)}{A} + (\omega_0\cdot\rho-R)\cdot\left(1-R^n\right)}. \quad (3.42)$$

Граничне значення коефіцієнта використання функціонального пристрою при $n \to \infty$ буде

$$\lim_{n\to\infty} H_{ФП} = \frac{R + (\omega_0\cdot\rho-R)}{1+\frac{(\rho-R)\cdot(1-R)}{A} + (\omega_0\cdot\rho-R)}, \quad (3.43)$$

$$\lim_{n\to\infty} H_{ФП} = \frac{\omega_0\cdot\rho}{1+\frac{(\rho-R)\cdot(1-R)}{A} + (\omega_0\cdot\rho-R)}, \quad (3.44)$$

$$\frac{\lim\limits_{n\to\infty} H_{ФП}}{\lim\limits_{n\to\infty} E_{ПК}} = \omega_o * \rho.* \quad (3.45)$$

3.7. Визначення діапазонів зміни параметрів моделі

Коефіцієнт використання функціонального пристрою є функцією таких змінних:

- Коефіцієнта навантаження ρ при виконанні програм без команд переходу, який визначається як структурними особливостями ядра, так і типом виконуваної програми. Він може набувати значень в широкому діапазоні:

- $\rho = 1$ означає, що продуктивності функціонального пристрою і ПК однакові, тобто, що система збалансована;

- $\rho \succ 1$ - ПК швидший ніж функціональний пристрій і система розбалансирована;

- $\rho \prec 1$ - функціональний пристрій швидший за ПК і система розбалансирована.

Вплив коефіцієнта навантаження ρ досліджується в діапазоні - 0.5; 1; 2.

- Частоти команд переходу - ω_n. Вплив ω_n досліджується в діапазоні 0,05; 0.10; 0.15; 0.20.

Частота команд переходу $\omega_П$=0.1 означає, що кожна десята команда у виконуваній програмі є командою переходу.

- Розміру буфера n. Вплив n досліджується в діапазоні 2; 4; 8; 16; 32.

- Коефіцієнт $A = \rho * \rho_n$ визначає співвідношення інтенсивностей перезавантаження конвеєра команд із стану bi в стан ai та інтенсивності виконання команд функціональним пристроєм.

Вплив A досліджується в діапазоні - 0.05; 0.10; 0.15; 0.20.

Аналіз виразу для коефіцієнта R показує, що $R \prec 1$ при будь-яких значеннях параметрів моделі.

3.8 Результати дослідження моделі ядра без блоку передбачення

В таблиці 3.1 приведені значення коефіцієнта R при різних значеннях коефіцієнта ρ і частоти команд переходу.

Таблиця 3.1 - Значення коэфіцієнта R при різних значеннях коефіцієнта ρ і частоти команд переходу

ρ		$\omega_П$					
		0.05	0.10	0.15	0.20	0.25	0.30
0.5	R	0.4542	0.4146	0.3792	0.3469	0.3170	0.2890
1	R	0.7764	0.6838	0.6127	0.5528	0.5	0.4523
2	R	0.9084	0.8292	0.7584	0.6938	0.6340	0.5780

Коефіцієнт навантаження R на функціональний пристрій при виконанні програм з командами переходу при будь-яких значеннях ρ і $\omega_П$>0 завжди менший 1.

Значення коефіцієнтів використання функціонального пристрою залежно від значення ρ, розміру буфера приведені в таблиці 3.2.

Таблиця 3.2 - Значення коефіцієнта використання функціонального пристрою залежно від значення ρ, розміру буфера n при $\omega_П = 0.20$; A=1.

ρ		n						
		2	4	8	16	32	64	∞
0.5	$H_{ФП}$	0.3402	0.3463	0.3469	0.3469	0.3469	0.3469	0.3469
1	$H_{ФП}$	0,5334	0,55	0,5527	0,5528	0,5528	0,5528	0,5528
2	$H_{ФП}$	0.6805	0.6926	0.6938	0.6938	0.6938	0.6938	0.6938

Оскільки при будь-яких значеннях ρ і $\omega_П$ R<1, то вже при розмірі буфера n=8 значень коефіцієнта використання функціонального пристрою дуже близькі до граничних значень. Похибка при n=8 не перевищує 0.5%, а при n=16, 32, 64 значення коефіцієнтів використання практично співпадають з граничними значеннями.

Отже, при проектуванні структури ядра мінімально необхідний розмір буфера має бути не менше 8 при різних значеннях коефіцієнта навантаження ρ. Проте, враховуючи, що ядро може виконувати різноманітні програми (в тому числі і такі, в яких дуже мало команд умовних переходів (тобто з $\omega_П$ близьким до нуля), то розмір буфера необхідно вибирати з урахуванням «найважчого» режиму роботи. А таким є режим з ρ=1 і $\omega_П$=0. Для такого режиму роботи близькі до граничних значень коефіцієнтів використання забезпечує буфер n=32. Такий розмір буфера задовольняє в усіх режимах роботи ядра.

Усі подальші дослідження проводитимуться в припущенні, що розмір буфера n=32 і при визначенні коефіцієнта використання пристроїв використовуватимуться граничні значення.

3.9 Дослідження впливу частоти команд переходів і коефіцієнта А

Залежності коефіцієнтів використання пристроїв від частоти команд переходів і коефіцієнта А при ρ = 1, n = 32 приведені в таблиці 3.3.

Таблиця 3.3. Залежності коефіцієнтів використання пристроїв від частоти команд переходів і коефіцієнта А при $\rho = 1$, n = 32.

A	$E_{ПК}$/ $H_{ФП}$	$\omega_п$			
		0,05	0,1	0,15	0,2
0.05	$E_{ПК}$	0.4601	0.3107	0.2360	0.1903
	$H_{ФП}$	0.4370	0.2799	0.2006	0.1525
0.10	$E_{ПК}$	0.5974	0.4512	0.3651	0.3080
	$H_{ФП}$	0.5676	0. 4061	0.3105	0.2464
0.25	$E_{ПК}$	0.7277	0.6188	0.5444	0.4885
	$H_{ФП}$	0.6915	0.5568	0.4626	0.3908
0.5	$E_{ПК}$	0.7881	0.7062	0.6506	0.6071
	$H_{ФП}$	0.7459	0.6355	0.5530	0.4857
1	$E_{ПК}$	0.8239	0.7596	0.7210	0.6911
	$H_{ФП}$	0.7764	0.6838	0.6127	0.5528
2	$E_{ПК}$	0.8343	0.7897	0.7621	0.7423
	$H_{ФП}$	0.7925	0.7108	0.6477	0.5938
	R	0.7764	0.6838	0.6127	0.5528

При збільшенні частоти команд переходів коефіцієнти використання пристроїв зменшуються і це зменшення може бути значним.

При збільшенні коефіцієнта А коефіцієнти використання пристроїв збільшуються, причому коефіцієнт збільшення росте зі збільшенням частоти команд переходів. Це явище пояснюється тим, що збільшення коефіцієнта А означає, що зменшується час входження системи в режим, що встановився, а, відповідно, зменшуються простої як пристрою керування, так і функціонального пристрою.

При ρ=1 незалежно від значення коефіцієнта А коэфіцієнт використання функціонального пристрою дорівнює R.

$$H_{ФП} = R = \frac{(1+\rho) - \sqrt{(1+\rho)^2 - 4*\omega_o * \rho}}{2} \quad . \qquad (3.4)$$

Висновки

Досліджено вплив різних параметрів моделі на її продуктивність при виконанні в ядрах процесорів реальних програм з командами умовних (та інших) переходів.

Розроблена модель ядра процесора при блокуванні пристрою керування при виявленні команди умовного переходу та інших команд переходу.

Встановлено, що при блокуванні пристрою керування при виявленні команди умовного переходу:

оптимальний розмір буфера для різних режимів роботи $n \geq 32$;

при розмірі буфера $n \geq 32$ для оцінки значень коефіцієнтів використання пристроїв можна використати їх граничні значення. Похибки при цьому значно менші 1%.

при збільшенні частоти команд переходів коефіцієнти використання пристроїв істотно зменшуються;

для досягнення більш високих показників продуктивності необхідно використати складніші методи компенсації негативного впливу команд переходів.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даного проекту бакалавра було «Дослідження ефективності ядер процесорів з спеціалізованими функціональними пристроями». Так як в процесі проектування виконувалось у домашніх умовах, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера на якому буде виконуватись дослідження.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці»[13] визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

4.1.1 Правові та організаційні основи охорони праці

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави, і особистої відповідальності працівників.

Державна політика в галузі охорони праці визначається відповідно до Конституції України Верховною Радою України і спрямована на створення належних, безпечних і здорових умов праці, запобігання нещасним випадкам та професійним захворюванням. Відповідно до статті 3 Закону України «Про охорону праці»[13] (далі –

Закону) законодавство про охорону праці складається з Закону, Кодексу законів про працю України, Закону України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності" та прийнятих відповідно до них нормативно-правових актів, норм міжнародного договору (ратифіковані Конвенції і Рекомендації МОТ, директиви Європейської Ради).

4.1.2 Організаційно-технічні заходи з безпеки праці

В організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 [17].

Обов’язковими вимогами враховане наступне:

– не слід допускати до роботи осіб, що в установленому порядку не пройшли навчання, інструктаж та перевірку знань з охорони праці, пожежної безпеки та цих Правил.

– на підприємстві/організації, де експлуатуються ЕОМ з відео дисплейними терміналами (ВДТ) і периферійними пристроями (ПП), розробляється інструкція з охорони праці відповідно до Положення про розробку інструкцій з охорони праці, затвердженого наказом Держнаглядохоронпраці від 29.01.98 N 9, зареєстрованого в Міністерстві юстиції України 07.04.98 за N 226/2666 [19].

– ознайомлення з правилами безпеки праці, одержання відповідних інструктажів засвідчується у журналі інструктажів.

4.2 Аналіз стану умов праці

Робота над створенням локальної комп'ютерної мережі проходить в побутовому приміщенні. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

4.2.1 Вимоги до приміщення

Геометричні розміри приміщення зазначені у таблиці 4.1.

Таблиця4.1 – розміри робочого місця

Параметр	Значення
Довжина, м	3
Ширина, м	5
Висота, м	2,5
Площа, $м^2$	15
Об'єм, $м^3$	37,5

Згідно до санітарних норм мікроклімату виробничих приміщень [19] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм – не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

4.2.2 Вимоги до організації робочого місця

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця [20] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 – Характеристика робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650

Продовження таблиці 4.2

Найменування параметра	Фактичне значення	Нормативне значення
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

У кабінеті є електрична мережа з напругою 220 В, яка створює небезпеку ураження електричним струмом. ПК та периферійні пристрої можуть бути джерелами електромагнітних випромінювань, аерозолів та шкідливих речовин (часток тонеру, оксидів нітрогену та озону).

За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет оснащений переносним вуглекислотним вогнегасником ВВК-5 .

Наявна аптечка для надання долікарської допомоги, а також у кабінеті роблять вологе прибирання та щоденно провітрюють приміщення.

4.2.3 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Ia), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до II – III ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви тривалістю 15 хв через кожну годину роботи;

4.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.3.1 Аналіз небезпечних та шкідливих факторів при розробці виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання , які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої. Основними робочими характеристиками персонального комп'ютера є:

- робоча напруга U = +220В +-5%;
- робочий струм I = 2А;
- споживана потужність P = 600 Вт.

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Нормативні документи
Фізичні		
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, серверного обладнання для роботи	[19.]
- підвищена або знижена вологість повітря	-//-	[19]
- підвищена або знижена рухливість повітря	-//-	[19]
- підвищений рівень напруги електричної мережі	-//-	[23] [25]
- підвищений рівень статичної електрики	-//-	[23]
- підвищена напруженість електромагнітного поля	-//-	[22]

Продовження таблиці 4.3

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Нормативні документи
- недостатність природного світла	порушення умов праці (вимог до приміщень)	[26]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	[26]
Психофізиологічні		
-нервово-психічна перевантаження	Розумова робота над проектом	[14] [20]
- фізичні (статичне – сидіння)	порушення умов праці та організації робочого часу	[14]

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 [20]. За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

4.3.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання. Небезпека загоряння пов'язана з особливістю комп'ютерів – із значною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах. При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

Для гасіння пожеж в офісному приміщенні пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними.

4.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт Іа. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають нормам [19] і наведені в табл. 4.4.

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура *C°*	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [19]. Рівні позитивних і негативних іонів у повітрі мають відповідати [19]. Для забезпечення оптимальних параметрів мікроклімату в приміщенні проводяться перерви в роботі співробітників, з метою його провітрювання. Існують спеціальні системи кондиціонування, які забезпечують підтримання в приміщенні балансу оптимальних параметрів мікроклімату.

Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.4.2 Освітлення

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Розрахунок освітлення

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n \qquad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 3 = 15$ м^2,

$S = 1/10 \cdot 15 = 1{,}5$ м^2.

Приймаємо 1 вікно площею S=1,5 м^2.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 3200 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (5.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \tag{4.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 15 м^2;

Z – поправочний коефіцієнт світильника (1,1 для люмінесцентних ламп);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 3200лм (для ЛБ-40-2).

Підставивши числові значення у формулу (5.2), отримуємо:

$$n = \frac{300 \cdot 15 \cdot 1{,}1 \cdot 1{,}5}{3200 \cdot 0.575 \cdot 2} = 2{,}018$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з 2-х люмінесцентних ламп загальною потужністю 40 Вт, напругою – 220 В.

4.4.3 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

- експлуатацію сертифікованого обладнання;

- дотримання заходів електробезпеки;

- забезпечення оптимальних параметрів мікроклімату;

- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об'єм приміщення 20 м^3, то потрібно подати не менш як 30 м^3/год повітря;

б) якщо об'єм приміщення у межах від 20 до 40 м^3, то потрібно подати не менш як 20 м^3/год повітря;

в) якщо об'єм приміщення становить понад 40 м^3, допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

- зниження рівня шуму та вібрації:

а) у джерелі виникнення, шляхом застосування раціональних конструкцій, нових матеріалів і технологічних процесів;

б) звукоізолювання устаткування за допомогою глушників, резонаторів, кожухів, захисних конструкцій, оздоблення стін, стелі, підлоги тощо;

в) використання засобів індивідуального захисту).

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;

- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

- не тягнути за мережевий кабель, щоб витягти вилку з розетки;

- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

- не залишати включені електроприлади без нагляду;

- не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

Від ураження струмом застосовують різні електричні захисні засоби:

а) Ізолюючі – ізолюють людини від струмоведучих або заземлених частин, а так-же від землі. Вони діляться на основні та додаткові.

б) Основні – володіють ізоляцією, здатної довго витримувати робоче напругу електроустановки і тому ними дозволяється стосуватися струмоведучих частин, знаходячи-трудящих під напругою.

в) Запобіжні – володіють ізоляцією нездатною витримати робоча напруга електроустановки, і тому вони не можуть самостійно захищати людину від ураження струмом під цим напругою. Їх значення - посилити захисні дії основних і ізолюючих засобів, разом з якими вони повинні застосовуватися, при чому при використання основних захисних засобів достатньо застосування одного запобіжного захисного засобу.

4.5.1 Розрахунок захисного заземлення

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом № [16], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_Д \cdot R_{пр.з.}}{R_{пр.з.-R_Д}}; \quad (4.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.} = R_д$.

Підставивши числові значення у формулу (5.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 Ом$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо $\rho = 40$ Ом·м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр.} = \psi \cdot \rho \quad (4.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в} = 1,7$ і горизонтальних $\rho_{розр.г} = 5,5$ Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 Ом \cdot м$$

$$\rho_{розр.в} = 5,5 \cdot 40 = 220 Ом \cdot м$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (4.).

$$R_B = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot 1_B} \cdot \left(\ln \frac{2 \cdot l_B}{d_{CT}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_B}{4 \cdot t - l_B} \right), \quad (4.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб – 2 − 3 м; $l_в$ = 3 м);

$d_{ст}$ – діаметр стержня (для труб – 0,03 − 0,05 м; $d_{ст}$ = 0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (5.6):

$$t = h_E + \frac{l_E}{2}, \tag{4.6}$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді

$$t = 0,8 + \frac{3}{2} = 2,3 \text{ м};$$

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

1) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2R_E}{R_Д} = \frac{2 \cdot 18,5}{4} = 9,25, \tag{4.7}$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в = 0,57$ (табличне значення).

2) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $\eta_в$, шт:

$$n = \frac{2 \cdot R_E}{R_Д \cdot \eta_E} = \frac{2 \cdot 18,5}{4 \cdot 0,57} \approx 16, \tag{4.8}$$

3) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_C = 1,05 \cdot L_B \cdot (n_B - 1), \tag{4.9}$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{ м}$$

Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_c} \cdot \ln \frac{2 \cdot l_c^2}{d_{см} \cdot h_г}, \quad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см} = 0{,}95b$, $b = 0{,}15$ м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

l_c – довжина з'єднувальної стрічки горизонтального заземлювача l_c, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0{,}95 \cdot 0{,}15 \cdot 0{,}5} = 8{,}1 \text{ Ом}$$

4) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с = 0{,}3$.

Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг.} = \frac{R_E \cdot R_г}{R_E \cdot \eta_c + R_г \cdot n_E \cdot \eta_E} \le R_д, \quad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг} < 4$ Ом, а саме:

$$R_{заг} = \frac{18{,}5 \cdot 8{,}1}{18{,}5 \cdot 0{,}3 + 8{,}1 \cdot 16 \cdot 0{,}57} = 1{,}9 \le R_д$$

При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

4.6 Екологія та охорона навколишнього середовища

Діяльність за темою магістерської роботи, а саме: Дослідження ефективності ядер процесорів з спеціалізованими функціональними пристроями в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

В процесі діяльності дослідження за допомогою ПК виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- Відпрацьовані люмінесцентні лампи - I клас небезпеки
- Акумулятор для джерел безперебійного живлення – III клас безпеки
- Змінні носії інформації - IV клас небезпеки
- Макулатура - IV клас небезпеки
- Побутові відходи - IV клас небезпеки

Зберігання відходів та їх утилізація виконується згідно до вимог Державних санітарних правил і норм ДСанПіН 2.2.7.029 [21].

4.7 Висновки до розілу 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і

потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

В магістерській роботі:

В розділі 1 розглянуті структури ядер сучасних процесорів на прикладі структури Haswel фірми Intel, їх основні блоки, організация та взаємодія. Розглянуті осоновні методи моделювання та проведено вибир прийнятних з них для створення моделей процесорів.

В розділі 2 розроблені аналітичні модели конвеєрної системи з універсальним функціональним пристроєм. Проведені дослідження системи з универсальним функціональним пристроєм в безперервному часі. Розроблені аналітичні модели конвеєрної системи з m спеціалізованими функціональними пристроями. Проведені дослідження створених моделей.

У розділі 3 розроблена аналітична модель ядра процесора з врахуванням впливу «перешкод» роботі конвеєра без блоку передбачення адрес переходів. Користуючись створеною моделлю було проведено дослідження.

У розділі 4 розглятнуті питання охорони праці та основні напрямки їх дотримання

ПЕРЕЛІК ВИКОРИСТОВАНОЇ ЛІТЕРАТУРИ

1. O. Bessonov, D. Fougere, B. Roux. Development of efficient computational kernels and linear algebra routines for out-of-order superscalar processors. Future Generation Computer Systems, V.21, No.5, 2005, pp.743-748.
2. M. Milenkovic, A. Milenkovic, J. Kulick. Demystifying Intel Branch Predictors. Proceedings of the Workshop on Duplicating, Deconstructing and Debunking, 2002.
3. IA-32 Intel Architecture Optimization Reference Manual. Intel, 2006.
4. С. Гарматюк. Современные десктопные процессоры архитектуры IA-32/64: общие принципы работы. iXBT.com, 2006.
5. B. Inkley. Inside the Intel Core Microarchitecture. Intel Developer Forum, 2006.
6. Коропів Ю. Імітаційне моделювання систем. Введення в моделювання з AnyLogic 5. - СПб.: БХВ-Петербург, 2005. - 400 с.
7. Михаил Кузьминский. Nehalem: структура и производительность. Открытые системы, №8, 2009.
8. Недзельский Д.А. Исследование эффективности одноядерных суперскалярных вычислительных систем. Луганск. Вісник Східноукраїнського національного університету ім. В. Даля. - 2011. - №15 (169) Ч. 2. - с. 133-140.
9. Посібник «Комп'ютерні системи»./ Уклад. Недзельський Д.О. – СНУ ім. В. Даля : 2017.- 305 с. Електронна форма
10. Орлов С.А., Цилькер Б.Я. Организация ЭВМ и систем. 2-е изд. – СПб.: Питер, 2011. – 688 с.
11. Брайант Р., О'Халларон Д. Компьютерные системы: архитектура и программирование. Пер. с англ. – СПб.: БХВ-Петербург, 2005. – 1104 с.: ил.
12. Клейнрок Л. Вычислительные системы с очередями. -М.: Мир, 1979. - 600с.
13. Закон України «Про охорону праці»;
14. НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин»;
15. НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці»;
16. НПАОП 0.00-4.15-98 «Положення про розробку інструкцій з охорони праці»;
17. НПАОП 40.1-1.01-97 «Правила безпечної експлуатації електроустановок»;

18. НАПБ Б.02.005-2003 Типове положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України;
19. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» ;
20. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»;
21. ДСанПіН 2.2.7.029 Гігієнічні вимоги щодо поводження з промисловими відходами та визначення їх класу небезпеки для здоров'я населення
22. ГОСТ 12.1.044-89 «ССБТ. Вогнестійкість. Номенклатура показників і методи їх визначення»;
23. ГОСТ 12.1.030-81 «Електробезпека. Захисне заземлення, занулення».
24. ГОСТ 12.1.006-84 «ССБТ. Електромагнітні поля радіочастот»;
25. ГОСТ 13109-97 «Електрична енергія. Сумісність технічних засобів. Норми якості електричної енергії в системах електропостачання загального призначення»;
26. ДБН В.2.5-28:2015 «Державні Будівельні Норми України. Природне і штучне освітлення»;
27. НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою».

ДОДАТОК А

Опис програм, використаних для проведення досліджень

1. Імітаційна модель системи із загальним буфером заявок (без урахування "перешкод" роботі конвеєра).

Програма призначена для проведення досліджень системи із загальним буфером заявок без урахування умовних переходів і інформаційних залежностей. Реалізована можливість задавати довільну кількість пристроїв керування і спеціалізованих функціональних пристроїв, довільні затримки на генерацію і виконання команд, об'єм загального буфера заявок, а також частоти появи команд різних типів.

Головне вікно програми приведено на рисунку А.1.1

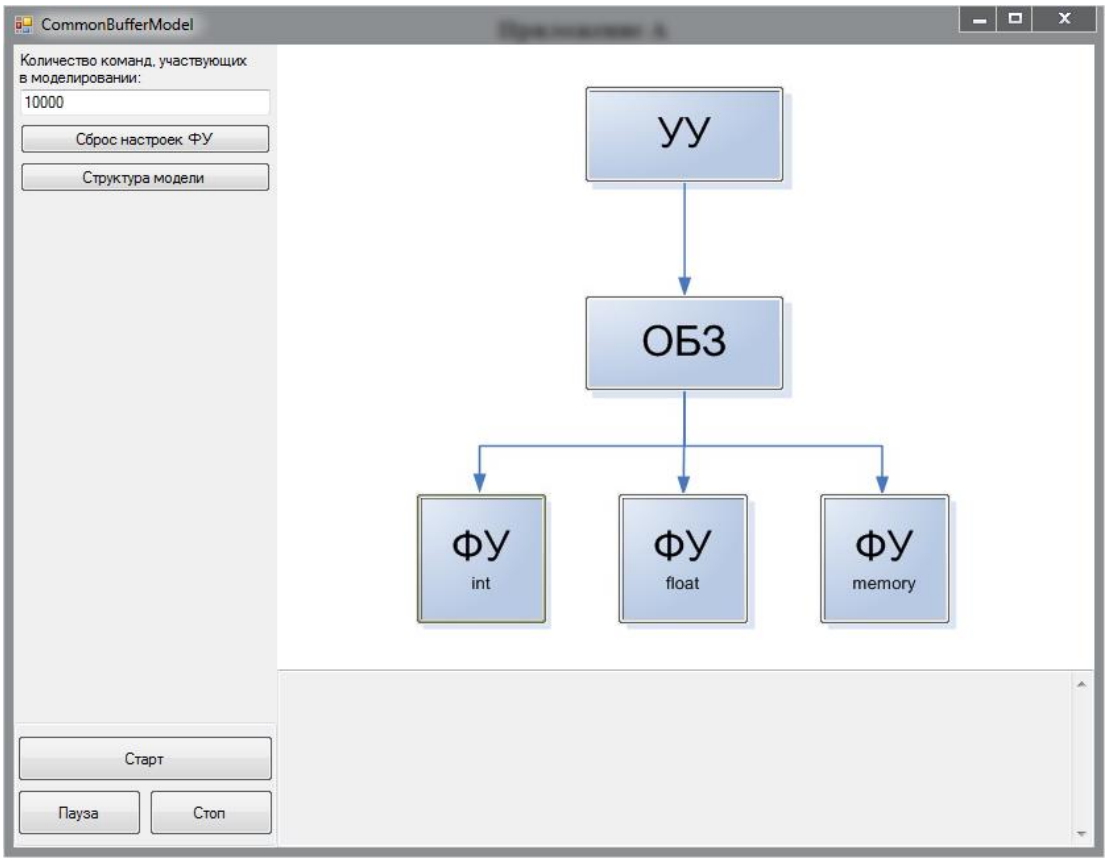


Рис. А.1.1 - Головне вікно програми

Основні характеристики, що стосуються роботи усієї моделі, задаються безпосередньо в головному вікні. Для завдання параметрів певного блоку системи (ПК, ФП, ЗБЗ) необхідно виконати натиснення на відповідну кнопку на головній формі. В результаті будуть відкриті наступні діалогові вікна:

а) для пристрою керування

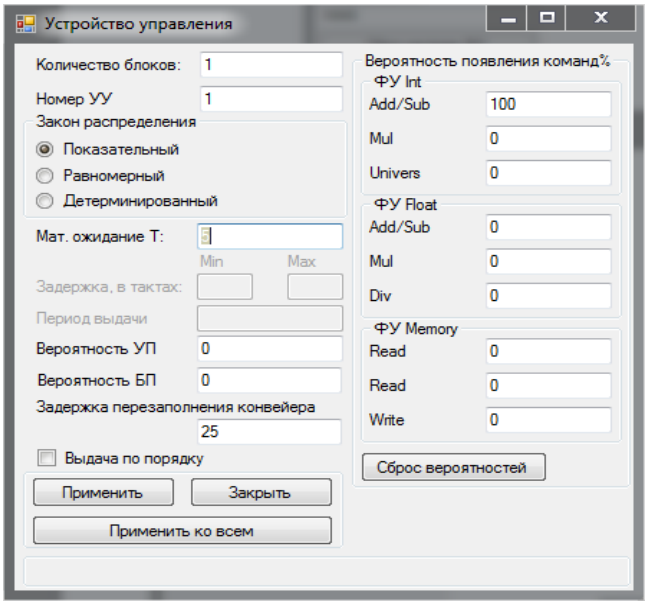


Рис А.1.2 - Діалогове вікно для завдання параметрів ПК

б) для загального буфера зявок

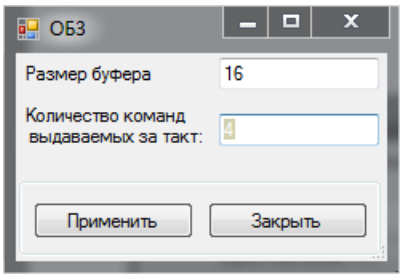


Рис А.1.3 - Діалогове вікно завдання параметрів ЗБЗ

в) для функціональних пристроїв

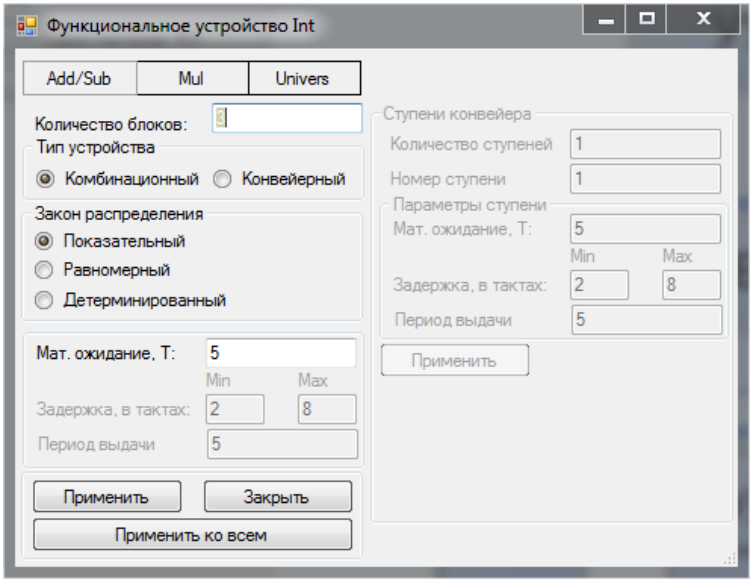
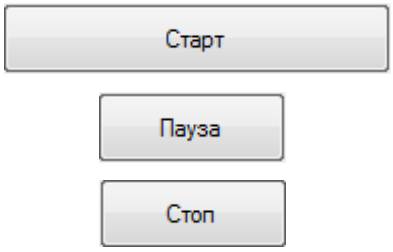


Рис А.1.4 - Діалогове вікно завдання параметрів ФП

Керування роботою ситемы здійснюється наступними кнопками:



Запустити модель на виконання

Припинити процес виконання

Перервати процес виконання і видати результат

Результати роботи моделі приведені на рисунку А.1.5 і відображають основні характеристики головних блоків системи (коефіцієнт використання, час зайнятості, час простою і так далі).

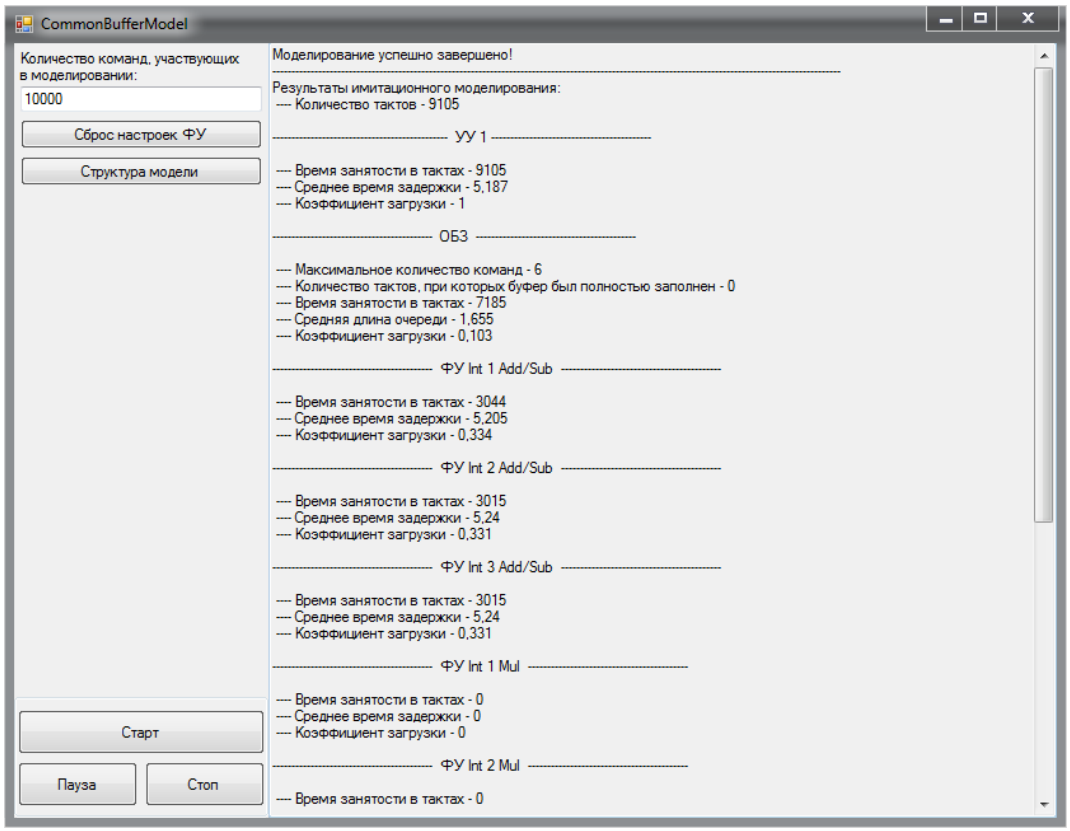


Рис А.1.5 - Результати роботи моделі

2. Імітаційна модель системи з індивідуальними буферами (без урахування "перешкод" роботі конвеєра).

Програма призначена для проведення досліджень системи з індивіуальними буферами заявок без урахування умовних переходів та інформаційних залежностей. Основна відмінність цієї моделі від приведеної в пункті 1 додатку А - наявність

індивідуального буфера заявок для кожного спеціалізованого функціонального пристрою керування і можливість завдання параметрів для кожного з них окремо.

Головне вікно програми приведено на рисунку А.2.1

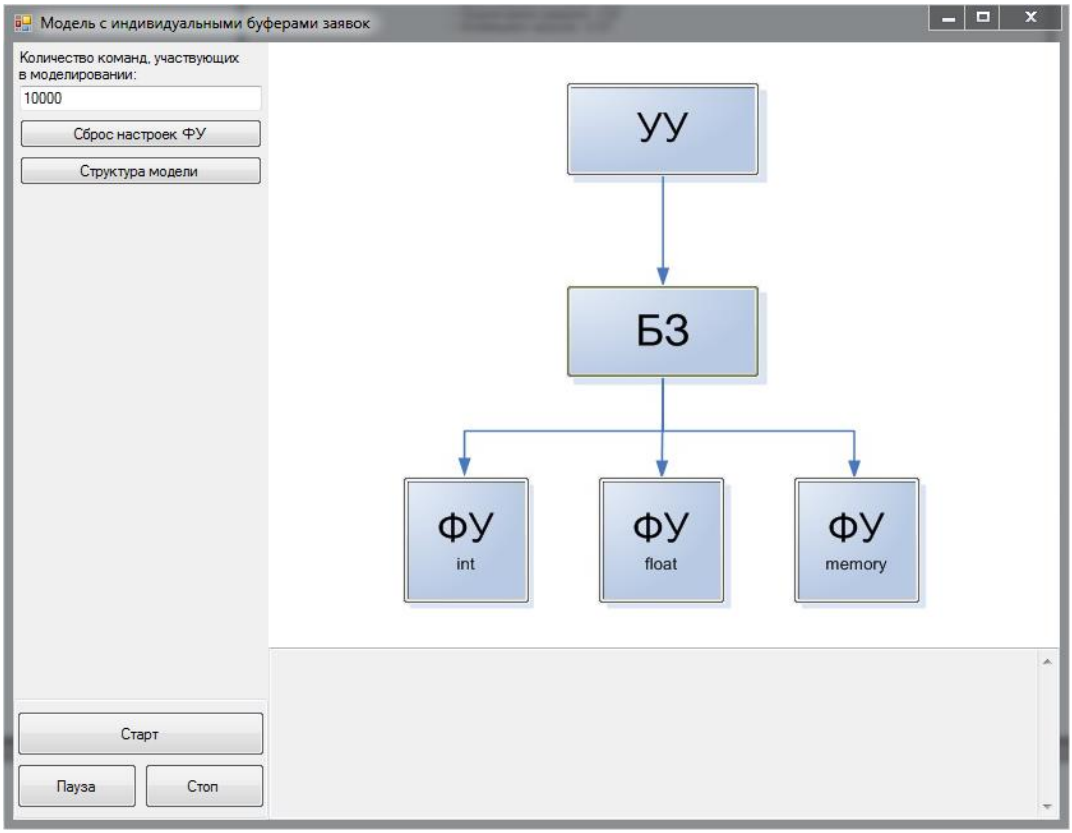


Рис. А.2.1 - Головне вікно програми

Опис використання моделі в пункті 1 додатку А справедливий і для цієї моделі, за винятком буфера заявок. Для завдання параметрів індивідуальних буферів заявок необхідно натиснути на кнопку "БЗ". Внаслідок буде виведене наступне діалогове вікно:

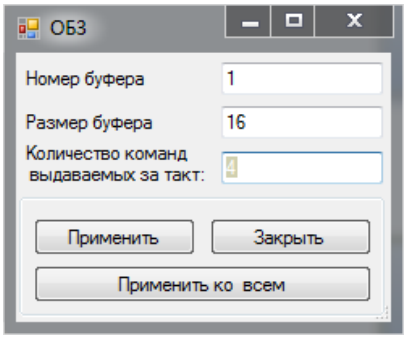


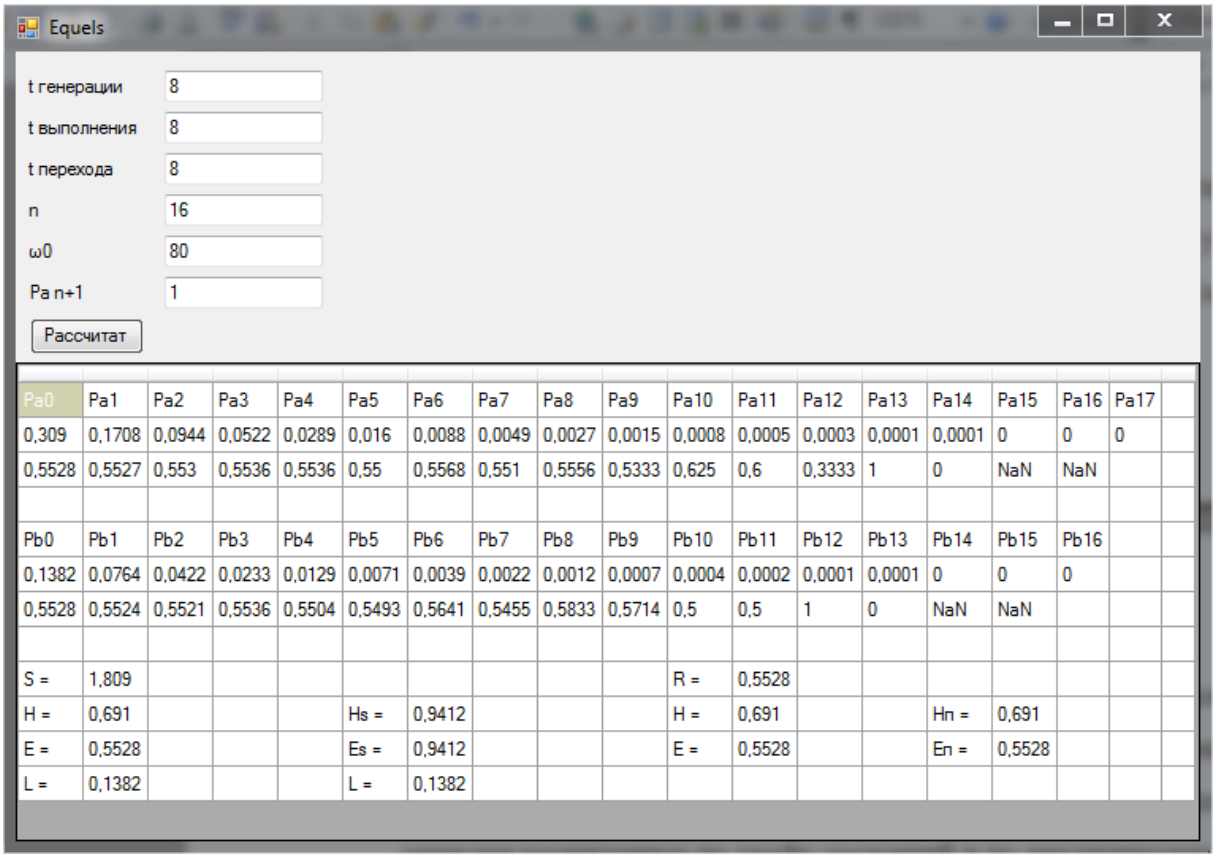
Рис А.2.2 - Діалогове вікно завдання параметрів БЗ

Виведення результатів аналогічне приведеному в пункті 1 додатку А.

3. Програма розрахунку коефіцієнтів використання системи з урахуванням умовних переходів (без блоку передбачення переходів)

Програма призначена для розрахунку коефіцієнтів використання пристрою керування і функціональних пристроїв по графові станів системи з урахуванням умовних переходів і циклів. У програмі реалізована можливість задавати період генерації, виконання команд обробки і переходу, розмір буфера заявок, частоту появи команд обробки і переходу. В результаті роботи програми розраховується вірогідність знаходження в кожному стані графа переходів, а також коефіцієнти використання розраховані по графові станів і по аналітичних формулах.

Головне вікно програми приведене на рисунку А.3.1



Pa0	Pa1	Pa2	Pa3	Pa4	Pa5	Pa6	Pa7	Pa8	Pa9	Pa10	Pa11	Pa12	Pa13	Pa14	Pa15	Pa16	Pa17	
0,309	0,1708	0,0944	0,0522	0,0289	0,016	0,0088	0,0049	0,0027	0,0015	0,0008	0,0005	0,0003	0,0001	0,0001	0	0	0	
0,5528	0,5527	0,553	0,5536	0,5536	0,55	0,5568	0,551	0,5556	0,5333	0,625	0,6	0,3333	1	0	NaN	NaN		
Pb0	Pb1	Pb2	Pb3	Pb4	Pb5	Pb6	Pb7	Pb8	Pb9	Pb10	Pb11	Pb12	Pb13	Pb14	Pb15	Pb16		
0,1382	0,0764	0,0422	0,0233	0,0129	0,0071	0,0039	0,0022	0,0012	0,0007	0,0004	0,0002	0,0001	0,0001	0	0	0		
0,5528	0,5524	0,5521	0,5536	0,5504	0,5493	0,5641	0,5455	0,5833	0,5714	0,5	0,5	1	0	NaN	NaN			
S =	1,809									R =	0,5528							
H =	0,691				Hs =	0,9412				H =	0,691			Hn =	0,691			
E =	0,5528				Es =	0,9412				E =	0,5528			En =	0,5528			
L =	0,1382				L =	0,1382												

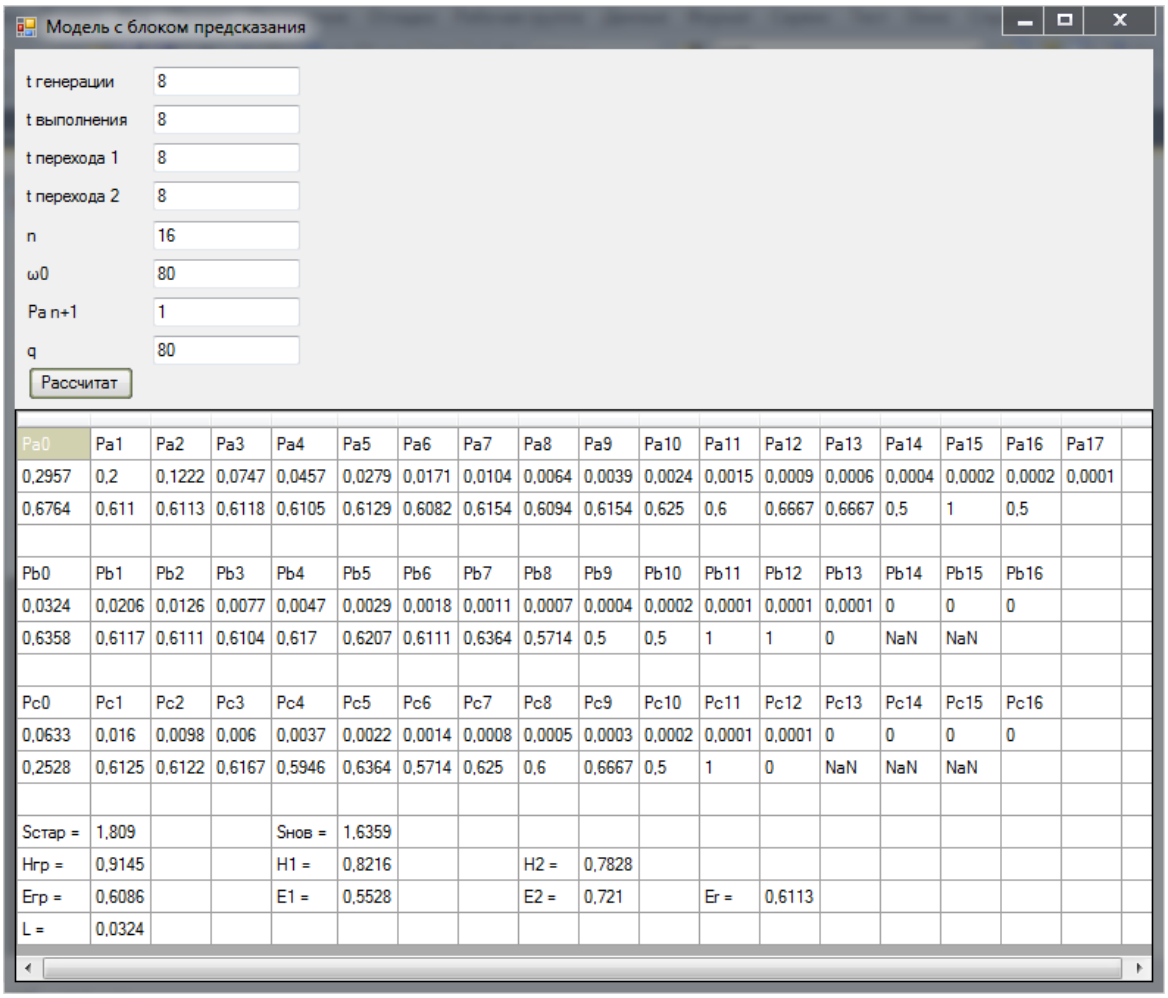
Рис. А.3.1 - Головне вікно програми

Запуск програми на виконання здійснюється натисненням на кнопку "Розрахувати".

4. Програма розрахунку коефіцієнтів використання системи з урахуванням умовних переходів (з блоком передбачення переходів)

Програма призначена для розрахунку коефіцієнтів використання пристрою керування і функціональних пристроїв по графові станів системи з урахуванням умовних переходів і циклів. У програмі реалізована можливість задавати період генерації, виконання команд обробки, команд переходу у разі вдалого і невдалого передбачення переходів блокомпередбачення, вірогідність вдалого передбачення переходу, розмір буфера заявок, частоту появи команд обробки і переходу. В результаті роботи програми розраховується вірогідність знаходження в кожному стані графа переходів, а також коефіцієнти використання розраховані по графові станів і по аналітичних формулах.

Головне вікно програми приведене на рисунку А.4.1



Pa0	Pa1	Pa2	Pa3	Pa4	Pa5	Pa6	Pa7	Pa8	Pa9	Pa10	Pa11	Pa12	Pa13	Pa14	Pa15	Pa16	Pa17
0,2957	0,2	0,1222	0,0747	0,0457	0,0279	0,0171	0,0104	0,0064	0,0039	0,0024	0,0015	0,0009	0,0006	0,0004	0,0002	0,0002	0,0001
0,6764	0,611	0,6113	0,6118	0,6105	0,6129	0,6082	0,6154	0,6094	0,6154	0,625	0,6	0,6667	0,6667	0,5	1	0,5	

Pb0	Pb1	Pb2	Pb3	Pb4	Pb5	Pb6	Pb7	Pb8	Pb9	Pb10	Pb11	Pb12	Pb13	Pb14	Pb15	Pb16
0,0324	0,0206	0,0126	0,0077	0,0047	0,0029	0,0018	0,0011	0,0007	0,0004	0,0002	0,0001	0,0001	0,0001	0	0	0
0,6358	0,6117	0,6111	0,6104	0,617	0,6207	0,6111	0,6364	0,5714	0,5	0,5	1	1	0	NaN	NaN	

Pc0	Pc1	Pc2	Pc3	Pc4	Pc5	Pc6	Pc7	Pc8	Pc9	Pc10	Pc11	Pc12	Pc13	Pc14	Pc15	Pc16
0,0633	0,016	0,0098	0,006	0,0037	0,0022	0,0014	0,0008	0,0005	0,0003	0,0002	0,0001	0,0001	0	0	0	0
0,2528	0,6125	0,6122	0,6167	0,5946	0,6364	0,5714	0,625	0,6	0,6667	0,5	1	0	NaN	NaN	NaN	

Sстар =	1,809			Sнов =	1,6359												
Hгр =	0,9145			H1 =	0,8216			H2 =	0,7828								
Eгр =	0,6086			E1 =	0,5528			E2 =	0,721		Er =	0,6113					
L =	0,0324																

Рис. А.4.1 - Головне вікно програми

Запуск програми на виконання здійснюється натисненням на кнопку "Розрахувати".

ДОДАТОК Б

Лістинг програми

Файл *Form1.cs:*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using model1.Models;
using model1.Blocks;
using model1.Instructions;

namespace model1
{

    public partial class Form1 : Form
    {
        const int FUTypeCount = 3;
        const int SubTypeCount = 3;
        public int SubTypeC { get { return SubTypeCount; } }
        public int FUTypeC { get { return FUTypeCount; } }
        public FU_Params[,] fu_params = new FU_Params[FUTypeCount,SubTypeCount];
        public CRB_Params crb_params = new CRB_Params();
        public MU_Params mu_params;
        protected FormFU formFU;
        protected FormCRB formCRB;
        protected FormMU formMU;
        protected Scheme formSch;

        protected bool MismatchLaw = false;
        protected bool resizeTrigger = false;

        protected Model activeModel = null;
        //Pre-Options
        //public int FUUniversalExp = 0;

        public int InstructionCount = 0;
        public int intensity = 1;
        //Post-Options
        protected int TicksCount = 0;//Кол-во тактов
        //CRB

        public Form1()
        {
            InitializeComponent();
            for (int i = 0; i < FUTypeCount; i++)
            {
                for (int j = 0; j < SubTypeCount; j++)
                {
                    fu_params[i,j] = new FU_Params();
                }
            }
            activeModel = new BaseModel(this);
            textBox22.Hide();
            if (activeModel == null)
            {
                return;
            }
            activeModel.SimulationCompleted += end_event;
            activeModel.InstructionEvent += progress_event;
            activeModel.EventInvoker = this;
            mu_params = new MU_Params(this);
            // activeModel.Configure();
            //Начальные установки на форме
            textBox8.Text = "10000";

            fu_params[0, 0].FUCount = 3;
            fu_params[0, 1].FUCount = 4;
            fu_params[0, 2].FUCount = 2;
```

```
            //Заполнение имен ФУ
            fu_params[0, 0].name = "ФУ Int";
            fu_params[0, 0].typeName = "Add/Sub";
            fu_params[0, 1].name = "ФУ Int";
            fu_params[0, 1].typeName = "Mul";
            fu_params[0, 2].name = "ФУ Int";
            fu_params[0, 2].typeName = "Univers";

            fu_params[1, 0].name = "ФУ Float";
            fu_params[1, 0].typeName = "Add/Sub";
            fu_params[1, 1].name = "ФУ Float";
            fu_params[1, 1].typeName = "Mul";
            fu_params[1, 2].name = "ФУ Float";
            fu_params[1, 2].typeName = "Div";

            fu_params[2, 0].name = "ФУ Memory";
            fu_params[2, 0].typeName = "Read";
            fu_params[2, 1].name = "ФУ Memory";
            fu_params[2, 1].typeName = "Read";
            fu_params[2, 2].name = "ФУ Memory";
            fu_params[2, 2].typeName = "Write";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            bool err = false;

            err = CheckMUExp();
            if (textBox8.Text == "0")
            {
                MessageBox.Show("Количество выполняемых команд не может быть нулем!");
                err = true;
            }

                /*for (int j = 0; j < SubTypeC; j++)
                {
                    if (fu_params[0, j].law != mu_params.law && fu_params[0, j].FUCount != 0)
                    {
                        MessageBox.Show("ФУ      Int      и      имеют      разные      законы
распределения!\r\nКоэффициенты загрузки системы не будут расчитаны.");
                        MismatchLaw = true;
                    }
                    else if (fu_params[1, j].law != mu_params.law && fu_params[1, j].FUCount !=
0)
                    {
                        MessageBox.Show("ФУ      Float      и      имеют      разные      законы
распределения!\r\nКоэффициенты загрузки системы не будут расчитаны.");
                        MismatchLaw = true;
                    }
                    else if (fu_params[2, j].law != mu_params.law && fu_params[2, j].FUCount !=
0)
                    {
                        MessageBox.Show("ФУ      Memory      и      имеют      разные      законы
распределения!\r\nКоэффициенты загрузки системы не будут расчитаны.");
                        MismatchLaw = true;
                    }
                    else if (fu_params[3, j].law != mu_params.law && fu_params[3, j].FUCount !=
0)
                    {
                        MessageBox.Show("ФУ     Universal     и     имеют     разные     законы
распределения!\r\nКоэффициенты загрузки системы не будут расчитаны.");
                        MismatchLaw = true;
                    }
                }*/

            if (!err)
            {
                textBox22.Show();
                //Reset actions
                if (textBox1.Text != "")
                {
                    activeModel.Reset();
                    textBox1.Clear();
                    TicksCount = 0;
                    resizeTrigger = false;
                    MismatchLaw = false;
```

```
                    }
                    InstructionCount = Convert.ToInt32(textBox8.Text);
                    //FUUniversalExp = Convert.ToInt32(textBox15.Text);
                    activeModel.Configure();
                    activeModel.SimulationStart();
                }
            }

            public bool CheckMUExp()
            {
                for (int i = 0; i < mu_params.MUCount; i++)
                {
                    int exp = mu_params.MUList[i].FUFloatASExp + mu_params.MUList[i].FUFloatDExp +
mu_params.MUList[i].FUFloatMExp + mu_params.MUList[i].FUIntASExp + mu_params.MUList[i].FUIntMExp +
mu_params.MUList[i].FUIntUExp + mu_params.MUList[i].FUMemoryR1Exp + mu_params.MUList[i].FUMemoryR2Exp +
mu_params.MUList[i].FUMemoryWExp;
                    if (exp != 100)
                    {
                        MessageBox.Show(String.Format("Некорректное значения вероятностей
появленпия команд в УУ №{0}: {1}",i+1,exp));
                        return true;
                    }
                }
                return false;
            }

            public void Show_Message(string message)
            {
                MessageBox.Show(message);
            }

            public void end_event(Model model)
            {
                textBox22.Hide();


                textBox1.AppendText("Моделирование успешно завершено!\r\n");
                textBox1.AppendText("-----------------------------------------------------------------
-----------------------------------------------------------------------\r\n");
                textBox1.AppendText("Результаты имитационного моделирования:\r\n");
                textBox1.AppendText(" ---- Количество тактов - "+TicksCount+"\r\n");

                for (int i = 0; i < mu_params.MUCount; i++)
                {
                    ManagerUnit mu = (ManagerUnit)model.Registry.GetBlock("MU"+(i+1).ToString());
                    textBox1.AppendText("\r\n---------------------------------------------      УУ
"+(i+1).ToString()+" ----------------------------------------\r\n\r\n");
                    textBox1.AppendText(" ---- Время занятости в тактах - " + mu.BusyTime +
"\r\n");
                    textBox1.AppendText(" ---- Среднее время задержки - " + Math.Round(mu.SrArph(),
3, MidpointRounding.ToEven).ToString() + "\r\n");
                    textBox1.AppendText("     ----     Коэффициент     загрузки     -     "     +
Math.Round(mu.LoadingFactor(), 3, MidpointRounding.ToEven).ToString() + "\r\n");

                    CommonRequestBuffer                              crb                              =
(CommonRequestBuffer)model.Registry.GetBlock("CRB"+(i+1).ToString());
                    textBox1.AppendText("\r\n-----------------------------------------         ОБЗ
"+(i+1).ToString()+"----------------------------------------\r\n\r\n");
                    textBox1.AppendText("    ----    Максимальное    количество    команд    -    "    +
crb.CRBMaxCount.ToString() + "\r\n");
                    textBox1.AppendText(" ---- Количество тактов, при которых буфер был полностью
заполнен - " + crb.CRBFullTicks.ToString() + "\r\n");
                    textBox1.AppendText("    ----    Время    занятости    в    тактах    -    "    +
crb.BusyTime.ToString() + "\r\n");
                    textBox1.AppendText("     ----     Средняя     длина     очереди     -     "     +
Math.Round(crb.SrQueue(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                    textBox1.AppendText(" ---- Коэффициент загрузки - " + Math.Round(crb.KPD(), 3,
MidpointRounding.ToEven).ToString() + "\r\n");
                }

                for (int j = 0; j < SubTypeC; j++)
                {
                    for (int i = 0; i < fu_params[0, j].FUCount; i++)
                    {
                        FunctionalUnit fu = (FunctionalUnit)model.Registry.GetBlock("FUI" + (j +
1).ToString() + (i + 1).ToString());
                        switch (j)
```

```
                            {
                                case 0:
                                    textBox1.AppendText("\r\n----------------------------------------
ФУ Int " + (i + 1).ToString() + " Add/Sub" + "  ----------------------------------------\r\n\r\n");
                                    break;
                                case 1:
                                    textBox1.AppendText("\r\n----------------------------------------
ФУ Int " + (i + 1).ToString() + " Mul" + "  ----------------------------------------\r\n\r\n");
                                    break;
                                case 2:
                                    textBox1.AppendText("\r\n----------------------------------------
ФУ Int " + (i + 1).ToString() + " Univers" + "  ----------------------------------------\r\n\r\n");
                                    break;
                            }
                            if (!fu_params[0, j].type)
                            {
                                textBox1.AppendText(" ---- Время занятости в тактах - " + fu.BusyTime +
"\r\n");
                                textBox1.AppendText("   ----   Среднее   время   задержки   -   "   +
Math.Round(fu.SrArph(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                                textBox1.AppendText("    ----    Коэффициент    загрузки    -    "    +
Math.Round(fu.LoadingFactor(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                            }
                            else
                            {
                                int l = 0;
                                for ( int k = fu_params[0, j].LevelCount-1; k >= 0 ; k--,l++)
                                {
                                    textBox1.AppendText("\r\n----------------------------------------
------------  Ступень №" + (l + 1).ToString() + "  ----------------------------------------\r\n");
                                    textBox1.AppendText("------------------ Время занятости в тактах -
" + fu.GetLevelBusyTime(k) + "\r\n");
                                    textBox1.AppendText("------------------ Среднее время задержки - "
+ Math.Round(fu.GetLevelSrArph(k), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                                    textBox1.AppendText("------------------ Коэффициент загрузки - " +
Math.Round(fu.GetLevelLoadingFactor(k), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                                }
                            }
                        }
                    }
                    for (int j = 0; j < SubTypeC; j++)
                    {
                        for (int i = 0; i < fu_params[1, j].FUCount; i++)
                        {
                            FunctionalUnit fu = (FunctionalUnit)model.Registry.GetBlock("FUF" + (j +
1).ToString() + (i + 1).ToString());
                            switch (j)
                            {
                                case 0:
                                    textBox1.AppendText("\r\n----------------------------------------
ФУ Float " + (i + 1).ToString() + " Add/Sub" + "  ----------------------------------------\r\n\r\n");
                                    break;
                                case 1:
                                    textBox1.AppendText("\r\n----------------------------------------
ФУ Float " + (i + 1).ToString() + " Mul" + "  ----------------------------------------\r\n\r\n");
                                    break;
                                case 2:
                                    textBox1.AppendText("\r\n----------------------------------------
ФУ Float " + (i + 1).ToString() + " Div" + "  ----------------------------------------\r\n\r\n");
                                    break;
                            }
                            if (!fu_params[1, j].type)
                            {
                                textBox1.AppendText(" ---- Время занятости в тактах - " + fu.BusyTime +
"\r\n");
                                textBox1.AppendText("   ----   Среднее   время   задержки   -   "   +
Math.Round(fu.SrArph(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                                textBox1.AppendText("    ----    Коэффициент    загрузки    -    "    +
Math.Round(fu.LoadingFactor(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                            }
                            else
                            {
                                int l = 0;
                                for (int k = fu_params[1, j].LevelCount - 1; k >= 0; k--, l++)
                                {
                                    textBox1.AppendText("\r\n----------------------------------------
------------  Ступень №" + (l + 1).ToString() + "  ----------------------------------------\r\n");
```

```
                                textBox1.AppendText("------------------ Время занятости в тактах -
" + fu.GetLevelBusyTime(k) + "\r\n");
                                textBox1.AppendText("------------------ Среднее время задержки - "
+ Math.Round(fu.GetLevelSrArph(k), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                                textBox1.AppendText("------------------ Коэффициент загрузки - " +
Math.Round(fu.GetLevelLoadingFactor(k), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                            }
                        }
                    }
                }
                for (int j = 0; j < SubTypeC; j++)
                {
                    for (int i = 0; i < fu_params[2, j].FUCount; i++)
                    {
                        FunctionalUnit fu = (FunctionalUnit)model.Registry.GetBlock("FUM" + (j +
1).ToString() + (i + 1).ToString());
                        //textBox1.AppendText("\r\n----------------------------------------    ФУ
Memory " + (i + 1).ToString() + (j + 1).ToString() + "   ----------------------------------------
\r\n\r\n");
                        switch (j)
                        {
                            case 0:
                                textBox1.AppendText("\r\n----------------------------------------
ФУ Memory " + (i + 1).ToString() + " Read" + "  ----------------------------------------\r\n\r\n");
                                break;
                            case 1:
                                textBox1.AppendText("\r\n----------------------------------------
ФУ Memory " + (i + 1).ToString() + " Read" + "  ----------------------------------------\r\n\r\n");
                                break;
                            case 2:
                                textBox1.AppendText("\r\n----------------------------------------
ФУ Memory " + (i + 1).ToString() + " Write" + "  ----------------------------------------\r\n\r\n");
                                break;
                        }
                        if (!fu_params[2, j].type)
                        {
                            textBox1.AppendText(" ---- Время занятости в тактах - " + fu.BusyTime +
"\r\n");
                            textBox1.AppendText("    ----    Среднее    время    задержки    -    "    +
Math.Round(fu.SrArph(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                            textBox1.AppendText("    ----    Коэффициент    загрузки    -    "    +
Math.Round(fu.LoadingFactor(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                        }
                        else
                        {
                            int l = 0;
                            for (int k = fu_params[2, j].LevelCount - 1; k >= 0; k--, l++)
                            {
                                textBox1.AppendText("\r\n----------------------------------------
------------  Ступень №" + (l + 1).ToString() + "  ----------------------------------------\r\n");
                                textBox1.AppendText("------------------ Время занятости в тактах -
" + fu.GetLevelBusyTime(k) + "\r\n");
                                textBox1.AppendText("------------------ Среднее время задержки - "
+ Math.Round(fu.GetLevelSrArph(k), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                                textBox1.AppendText("------------------ Коэффициент загрузки - " +
Math.Round(fu.GetLevelLoadingFactor(k), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                            }
                        }
                    }
                }
                /*for (int j = 0; j < SubTypeC; j++)
                {
                    for (int i = 0; i < fu_params[3, j].FUCount; i++)
                    {
                        FunctionalUnit fu = (FunctionalUnit)model.Registry.GetBlock("FUU" + (i +
1).ToString() + (j + 1).ToString());
                        textBox1.AppendText("\r\n----------------------------------------      ФУ
Universal " + (i + 1).ToString() + (j + 1).ToString() + "  ----------------------------------------
\r\n\r\n");
                        textBox1.AppendText(" ---- Время занятости в тактах - " + fu.BusyTime +
"\r\n");
                        textBox1.AppendText("    ----    Среднее    время    задержки    -    "    +
Math.Round(fu.SrArph(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                        textBox1.AppendText("     ----     Коэффициент     загрузки     -     "     +
Math.Round(fu.LoadingFactor(), 3, MidpointRounding.ToEven).ToString() + "\r\n");
                    }
                }*/
```

```
                    /* if (!MismatchLaw && mu_params.law)
                     {
                         textBox1.AppendText("-----------------------------------------------------------
-------------------------------------------------------------------------------------\r\n");
                         textBox1.AppendText("Результаты аналитического расчета:\r\n");
                         double p = Math.Round((fu_params[0,0].MathExp / mu_params.MathExp), 3,
MidpointRounding.ToEven);
                         double E, H;
                         textBox1.AppendText(" ---- p = " + p.ToString() + "\r\n");
                         if (p == 1)
                         {
                             double d = (double)(this.crb_params.Volume + 1) / (this.crb_params.Volume +
2);
                             E = H = Math.Round(d, 3, MidpointRounding.ToEven);
                             textBox1.AppendText(" ---- Коэффициент загрузки УУ (H) - " + H.ToString() +
"\r\n");
                             textBox1.AppendText(" ---- Коэффициент загрузки ФУ (E) - " + E.ToString() +
"\r\n");
                         }
                         else
                         {
                             E = Math.Round(((1 - Math.Pow(p, (crb_params.Volume + 1))) / (1 -
Math.Pow(p, (crb_params.Volume + 2))))*p, 3, MidpointRounding.ToEven);
                             H = Math.Round((1 - Math.Pow(p, (crb_params.Volume + 1))) / (1 -
Math.Pow(p, (crb_params.Volume + 2))), 3, MidpointRounding.ToEven);
                             textBox1.AppendText(" ---- Коэффициент загрузки УУ (H) - " + H.ToString() +
"\r\n");
                             textBox1.AppendText(" ---- Коэффициент загрузки ФУ (E) - " + E.ToString() +
"\r\n");
                         }
                     }*/

                }

                public void progress_event(Model model, string @event, Instruction instruction)
                {
                    textBox22.Text = TicksCount.ToString();
                    TicksCount++;
                }

                private void button2_Click(object sender, EventArgs e)
                {
                    if(activeModel.State != ModelState.Paused)
                        activeModel.SimulationPause();
                    else if(activeModel.State == ModelState.Paused)
                        activeModel.SimulationResume();
                }

                private void button3_Click(object sender, EventArgs e)
                {
                    /*if (activeModel.State != ModelState.Stopped)
                        activeModel.SimulationStop();
                    else*/
                        activeModel.SimulationEnd();
                }

                private void button4_Click(object sender, EventArgs e)
                {
                    formFU = new FormFU(this, 0);
                    string[] s=new string[3];
                    s[0]="Add/Sub";s[1]="Mul";s[2]="Univers";
                    formFU.Init(s);
                    formFU.ShowDialog();
                }

                private void button5_Click(object sender, EventArgs e)
                {
                    formFU = new FormFU(this, 1);
                    string[] s = new string[3];
                    s[0] = "Add/Sub"; s[1] = "Mul"; s[2] = "Univers";
                    formFU.Init(s);
                    formFU.ShowDialog();
                }

                private void button6_Click(object sender, EventArgs e)
                {
                    formFU = new FormFU(this, 2);
```

```
        string[] s = new string[3];
        s[0] = "Read"; s[1] = "Read"; s[2] = "Write";
        formFU.Init(s);
        formFU.ShowDialog();
    }

    private void button7_Click(object sender, EventArgs e)
    {
        formFU = new FormFU(this, 3);
        string[] s = new string[3];
        s[0] = "First"; s[1] = "Second"; s[2] = "Third";
        formFU.Init(s);
        formFU.ShowDialog();
    }

    private void textBox1_DoubleClick(object sender, EventArgs e)
    {
        if (!resizeTrigger)
        {
            /*while (textBox1.Height != 609)
            {
                textBox1.Height += 1;
                //textBox1.Refresh();
            }*/
            textBox1.Height = 610;
            resizeTrigger = true;
        }
        else
        {
            /*while (textBox1.Height != 135)
            {
                textBox1.Height -= 1;
            }*/
            textBox1.Height = 135;
            resizeTrigger = false;
        }
    }

    private void button8_Click(object sender, EventArgs e)
    {
        formCRB = new FormCRB(this);
        formCRB.ShowDialog();
    }

    private void button9_Click(object sender, EventArgs e)
    {
        formMU = new FormMU(this);
        formMU.ShowDialog();
    }

    private void button11_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < FUTypeCount; i++)
        {
            for (int j = 0; j < SubTypeCount; j++)
            {
                fu_params[i, j].FUCount = 0;
            }
        }
    }

    private void button10_Click(object sender, EventArgs e)
    {
        formSch = new Scheme(this);
        formSch.Show();
    }
}

//Хранит параметры ФУ
public class FU_Params
{
    public FU_Params()
    {
        LevelList.Add(new Level());
    }
    public int FUCount = 0;
    public int FUMin = 2;
```

```
    public int FUMax = 8;
    public double MathExp = 5;
    public int Period = 5;
    public int law = 1; //1 - показательный закон, 2 - равномерный, 3 - детерминированный
    public bool type = false;//true - конвейерный, false - комбинационный
    private int levelCount = 1;
    public string name = "";
    public string typeName = "";

    public int[] bufCount = { 4, 4, 4, 4 };
    public int[] exp = { 90, 95, 98, 100 };

    public int LevelCount
    {
        get { return levelCount; }
        set
        {
            if (levelCount < value)
            {
                while (LevelList.Count < value)
                {
                    LevelList.Add(new Level());
                }
            }
            else
            {
                while (LevelList.Count != value)
                {
                    LevelList.RemoveAt(LevelList.Count - 1);
                }
            }
            levelCount = value;
        }
    }
    public List<Level> LevelList = new List<Level>();
}

public class Level
{
    public int FUMin = 2;
    public int FUMax = 8;
    public double MathExp = 5;
    public int Period = 5;
}

//Хранит параметры ОБЗ
public class CRB_Params
{
    public CRB_Params()
    {
        CRBList.Add(new CRB());
    }

    public int CBRCount = 1;
    public List<CRB> CRBList = new List<CRB>();
}

public class CRB
{
    public int Volume = 16;
    public int CountIntInTick = 4;
}

//Хранит параметры УУ
public class MU_Params
{
    public MU_Params(Form1 form)
    {
        MUList.Add(new MU());
        this.form = form;
    }
    private int muCount = 1;
    private Form1 form;
    public  List<MU> MUList = new List<MU>();
    public int MUCount
    {
        get { return muCount; }
```

```
            set {
                if (muCount < value)
                {
                    while (MUList.Count < value)
                    {
                        MUList.Add(new MU());
                        form.crb_params.CRBList.Add(new CRB());
                    }
                }
                else
                {
                    while (MUList.Count != value)
                    {
                        MUList.RemoveAt(MUList.Count - 1);
                        form.crb_params.CRBList.RemoveAt(form.crb_params.CRBList.Count - 1);
                    }
                }
                muCount = value;
                form.crb_params.CBRCount = value;
                }
        }
    }

    public class MU
    {
        //Закон распределения
        public int FUMin = 2;
        public int FUMax = 8;
        public double MathExp = 5;
        public int Period = 5;
        //Время перезаполнения конвейера
        public double RefreshMathExp = 25;
        public int law = 1; //1 - показательный закон, 2 - равномерный, 3 - детерминированный
        //Для вероятности выдачи команд разных типов
        public int FUIntASExp = 100;
        public int FUIntMExp = 0;
        public int FUIntUExp = 0;
        public int FUFloatASExp = 0;
        public int FUFloatMExp = 0;
        public int FUFloatDExp = 0;
        public int FUMemoryR1Exp = 0;
        public int FUMemoryR2Exp = 0;
        public int FUMemoryWExp = 0;
        public bool SerialLaw = false;
        //Вероятность условного перехода
        public int ExpBranch = 0;
        public int ExpPB = 0;

    }
}
```

Файл *FormCRB.cs:*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace model1
{
    public partial class FormCRB : Form
    {
        Form1 mainForm;
        int index;

        public FormCRB(Form1 form)
        {
            InitializeComponent();
            this.mainForm = form;
            textBox1.Text = mainForm.crb_params.CRBList[index].Volume.ToString();
            textBox2.Text = mainForm.crb_params.CRBList[index].CountIntInTick.ToString();
            textBox3.Text = "1";
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "" && textBox2.Text != "")
            {
                mainForm.crb_params.CRBList[index].Volume = Convert.ToInt32(textBox1.Text);
                mainForm.crb_params.CRBList[index].CountIntInTick                                    =
Convert.ToInt32(textBox2.Text);
            }
            else
            {
                MessageBox.Show("Не все параметры указаны!");
            }
            this.Close();
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {
            if  (Convert.ToInt32(textBox3.Text)  >  0  &&  Convert.ToInt32(textBox3.Text)  <=
mainForm.crb_params.CBRCount)
            {
                index = Convert.ToInt32(textBox3.Text)-1;
            }
            else
            {
                MessageBox.Show(String.Format("Неверный  номер  буфера:  {0}.\nБуфер  с  таким
номером не существует!", Convert.ToInt32(textBox3.Text)));
                textBox3.Text = "1";
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < mainForm.crb_params.CBRCount; i++)
            {
                if (textBox1.Text != "" && textBox2.Text != "")
                {
                    mainForm.crb_params.CRBList[i].Volume = Convert.ToInt32(textBox1.Text);
                    mainForm.crb_params.CRBList[i].CountIntInTick                                    =
Convert.ToInt32(textBox2.Text);
                }
                else
                {
                    MessageBox.Show("Не все параметры указаны!");
```

```
                }
            }
            this.Close();
        }
    }
}
```

Файл *FormFU.cs:*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace model1
{
    public partial class FormFU : Form
    {
        Form1 mainForm;
        int index;
        int selectedIndex;
        int selectedLaw = 1;

        public FormFU(Form1 form, int index)
        {
            InitializeComponent();
            this.mainForm = form;
            this.index = index;
            textBox10.Text = "1";
            textBox11.Text = "1";
            if (index != 2)
            {
                button8.Hide();
            }
            else
            {
                button8.Show();
                for(int i=0;i<3;i++)
                {
                    mainForm.fu_params[2, i].type = true;
                    mainForm.fu_params[2, i].LevelCount = 4;
                    mainForm.fu_params[2, i].law = 3;
                    groupBox2.Enabled = false;
                    groupBox4.Enabled = false;
                    groupBox1.Enabled = false;
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            label2.Enabled = true;
            textBox2.Enabled = true;
            textBox2.Text = mainForm.fu_params[index, selectedIndex].MathExp.ToString();

            label11.Enabled = false;
            label3.Enabled = false;
            label4.Enabled = false;
            textBox6.Enabled = false;
            textBox7.Enabled = false;
            textBox3.Enabled = false;
            label5.Enabled = false;
//            textBox3.Text = "";
//            textBox6.Text = "";
//            textBox7.Text = "";

            label7.Enabled = true;
            textBox8.Enabled = true;
            textBox8.Text                          =                          mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-1].MathExp.ToString();

            label10.Enabled = false;
            label9.Enabled = false;
            label8.Enabled = false;
```

```
            textBox9.Enabled = false;
            textBox5.Enabled = false;
            textBox4.Enabled = false;
            label6.Enabled = false;

            selectedLaw = 1;
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            label11.Enabled = true;
            label3.Enabled = true;
            label4.Enabled = true;
            textBox6.Enabled = true;
            textBox6.Text = mainForm.fu_params[index, selectedIndex].FUMin.ToString();
            textBox7.Enabled = true;
            textBox7.Text = mainForm.fu_params[index, selectedIndex].FUMax.ToString();

            label2.Enabled = false;
            textBox2.Enabled = false;
            textBox3.Enabled = false;
            label5.Enabled = false;
//            textBox2.Text = "";
//            textBox3.Text = "";

            label10.Enabled = true;
            label9.Enabled = true;
            label8.Enabled = true;
            textBox9.Enabled = true;
            textBox9.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-1].FUMin.ToString();
            textBox5.Enabled = true;
            textBox5.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-1].FUMax.ToString();

            label7.Enabled = false;
            textBox8.Enabled = false;
            textBox4.Enabled = false;
            label6.Enabled = false;


            selectedLaw = 2;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            bool err = false;
            err = CheckForm();

            if (!err)
            {
                //Занесение полученный данных в структуры главной формы
                FillMainStruct(index,selectedIndex);
                //Красим батон
                switch (selectedIndex)
                {
                    case 0:
                        button5.BackColor = Color.GreenYellow;
                        break;
                    case 1:
                        button4.BackColor = Color.GreenYellow;
                        break;
                    case 2:
                        button3.BackColor = Color.GreenYellow;
                        break;
                }
                //Закрытие формы
                //this.Close();
            }

        }

        bool CheckForm()
        {
            bool err = false;
            //Проверка корректности ввода параметров
```

```
            if ((textBox2.Text != "" || (textBox6.Text != "" && textBox7.Text != "")) && textBox1.Text
!= "")
            {
                try
                {
                    int i = Convert.ToInt32(textBox1.Text);
                    if (i < 0)
                    {
                        MessageBox.Show("Количество блоков не может быть отрицательным!");
                        err = true;
                    }
                }
                catch (FormatException)
                {
                    MessageBox.Show("Введенное количество блоков имеет неверный формат!");
                    err = true;
                }

                if (radioButton1.Checked)
                {
                    try
                    {
                        if (Convert.ToInt32(textBox2.Text) < 0)
                        {
                            MessageBox.Show("Мат. ожидание меньше нуля!");
                            err = true;
                        }
                    }
                    catch (FormatException)
                    {
                        MessageBox.Show("Введенное значение мат. ожидания имеет неверный формат!");
                        err = true;
                    }
                }
                else if (radioButton2.Checked)
                {
                    try
                    {
                        if (Convert.ToInt32(textBox6.Text) < 0 || Convert.ToInt32(textBox7.Text) < 0)
                        {
                            MessageBox.Show("Задержка должна быть больше нуля!");
                            err = true;
                        }
                        if (Convert.ToInt32(textBox6.Text) > Convert.ToInt32(textBox7.Text))
                        {
                            MessageBox.Show("Минимальное значение задержки больше, чем максимальное!");
                            err = true;
                        }
                    }
                    catch (FormatException)
                    {
                        MessageBox.Show("Введенное значение задержки имеет неверный формат!");
                        err = true;
                    }
                }
            }
            else
            {
                MessageBox.Show("Не все параметры указаны!");
                err = true;
            }
            return err;
        }

        void FillMainStruct(int index, int selectedIndex)
        {
            mainForm.fu_params[index,selectedIndex].FUCount = Convert.ToInt32(textBox1.Text);
            if (radioButton1.Checked)
            {
                mainForm.fu_params[index, selectedIndex].law = 1;
                mainForm.fu_params[index, selectedIndex].MathExp = Convert.ToInt32(textBox2.Text);
            }
            else if (radioButton2.Checked)
            {
                mainForm.fu_params[index, selectedIndex].law = 2;
                mainForm.fu_params[index, selectedIndex].FUMin = Convert.ToInt32(textBox6.Text);
                mainForm.fu_params[index, selectedIndex].FUMax = Convert.ToInt32(textBox7.Text);
```

```
            }
            else if (radioButton5.Checked)
            {
                mainForm.fu_params[index, selectedIndex].law = 3;
                mainForm.fu_params[index, selectedIndex].Period = Convert.ToInt32(textBox3.Text);
            }

            mainForm.fu_params[index, selectedIndex].type = radioButton4.Checked;
        }

        public void Init(string[] s)
        {
            button5.Text = s[0];
            button4.Text = s[1];
            button3.Text = s[2];
            selectedIndex = 0;
            ButtonSelect(0);
        }

        private void ButtonSelect(int selIndex)
        {
            switch (selIndex)
            {
                case 0:
                    button5.FlatStyle = FlatStyle.Popup;
                    button4.FlatStyle = FlatStyle.Flat;
                    button3.FlatStyle = FlatStyle.Flat;
                    break;
                case 1:
                    button5.FlatStyle = FlatStyle.Flat;
                    button4.FlatStyle = FlatStyle.Popup;
                    button3.FlatStyle = FlatStyle.Flat;
                    break;
                case 2:
                    button5.FlatStyle = FlatStyle.Flat;
                    button4.FlatStyle = FlatStyle.Flat;
                    button3.FlatStyle = FlatStyle.Popup;
                    break;
            }
            switch (index)
            {
                case 0:
                    this.Text = "Функциональное устройство Int";
                    break;
                case 1:
                    this.Text = "Функциональное устройство Float";
                    break;
                case 2:
                    this.Text = "Функциональное устройство Memory";
                    break;
                case 3:
                    this.Text = "Функциональное устройство Universal";
                    break;
            }

            textBox1.Text = mainForm.fu_params[index, selectedIndex].FUCount.ToString();
            radioButton1.Checked = mainForm.fu_params[index, selectedIndex].law == 1 ? true : false;
            radioButton2.Checked = mainForm.fu_params[index, selectedIndex].law == 2 ? true : false;
            radioButton5.Checked = mainForm.fu_params[index, selectedIndex].law == 3 ? true : false;
            radioButton4.Checked = mainForm.fu_params[index, selectedIndex].type;
            radioButton3.Checked = !mainForm.fu_params[index, selectedIndex].type;
            textBox2.Text = mainForm.fu_params[index, selectedIndex].MathExp.ToString();
            textBox6.Text = mainForm.fu_params[index, selectedIndex].FUMin.ToString();
            textBox7.Text = mainForm.fu_params[index, selectedIndex].FUMax.ToString();
            textBox3.Text = mainForm.fu_params[index, selectedIndex].Period.ToString();

            textBox8.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].MathExp.ToString();
            textBox9.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].FUMin.ToString();
            textBox5.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].FUMax.ToString();
            textBox4.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].Period.ToString();

            textBox11.Text = "1";
            textBox10.Text = mainForm.fu_params[index, selectedIndex].LevelCount.ToString();
```

```
        }


        private void button5_Click(object sender, EventArgs e)
        {
            selectedIndex = 0;
            ButtonSelect(0);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            selectedIndex = 1;
            ButtonSelect(1);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            selectedIndex = 2;
            ButtonSelect(2);
        }

        private void radioButton5_CheckedChanged(object sender, EventArgs e)
        {
            textBox3.Enabled = true;
            textBox3.Text = mainForm.fu_params[index, selectedIndex].Period.ToString();
            label5.Enabled = true;

            label2.Enabled = false;
            textBox2.Enabled = false;
            label11.Enabled = false;
            label3.Enabled = false;
            label4.Enabled = false;
            textBox6.Enabled = false;
            textBox7.Enabled = false;
            textBox4.Enabled = true;
            textBox4.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-1].Period.ToString();
            label6.Enabled = true;

            label7.Enabled = false;
            textBox8.Enabled = false;
            label10.Enabled = false;
            label9.Enabled = false;
            label8.Enabled = false;
            textBox9.Enabled = false;
            textBox5.Enabled = false;

            selectedLaw = 3;
        }

        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            groupBox6.Enabled = true;
            if (selectedLaw == 1)
            {
                radioButton1.Checked = true;
            }
            else if (selectedLaw == 2)
            {
                radioButton2.Checked = true;
            }
            else if (selectedLaw == 3)
            {
                radioButton5.Checked = true;
            }
            groupBox4.Enabled = false;
        }

        private void radioButton4_CheckedChanged(object sender, EventArgs e)
        {
            if (index != 2)
            {
                groupBox4.Enabled = true;
                button8.Hide();
            }
            else
```

```
            {
                button8.Show();
            }
            if (selectedLaw == 1)
            {
                radioButton1.Checked = true;
            }
            else if (selectedLaw == 2)
            {
                radioButton2.Checked = true;
            }
            else if (selectedLaw == 3)
            {
                radioButton5.Checked = true;
            }

            groupBox6.Enabled = false;
        }

        private void UpdateLevel()
        {
            textBox8.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].MathExp.ToString();
            textBox9.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].FUMin.ToString();
            textBox5.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].FUMax.ToString();
            textBox4.Text                           =                           mainForm.fu_params[index,
selectedIndex].LevelList[Convert.ToInt32(textBox11.Text) - 1].Period.ToString();
        }

        private void textBox10_TextChanged(object sender, EventArgs e)
        {
            mainForm.fu_params[index, selectedIndex].LevelCount = Convert.ToInt32(textBox10.Text);
        }

        private void textBox11_TextChanged(object sender, EventArgs e)
        {
            if   (Convert.ToInt32(textBox11.Text)   >   0   &&   Convert.ToInt32(textBox11.Text)   <=
Convert.ToInt32(textBox10.Text))
            {
                UpdateLevel();
            }
            else
            {
                MessageBox.Show(String.Format("Ступени  конвейера  стаким  индексом  не  существует  :
{0}",Convert.ToInt32(textBox11.Text)));
                textBox11.Text = "1";
            }

        }

        private void button6_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                mainForm.fu_params[index,     selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-
1].MathExp = Convert.ToInt32(textBox8.Text);
            }
            else if (radioButton2.Checked)
            {
                mainForm.fu_params[index,     selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-
1].FUMin = Convert.ToInt32(textBox9.Text);
                mainForm.fu_params[index,     selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-
1].FUMax = Convert.ToInt32(textBox5.Text);
            }
            else if (radioButton5.Checked)
            {
                mainForm.fu_params[index,     selectedIndex].LevelList[Convert.ToInt32(textBox11.Text)-
1].Period = Convert.ToInt32(textBox4.Text);
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            bool err = false;
            err = CheckForm();
```

```
            if (!err)
            {
                for (int i = 0; i < 3; i++)
                {
                    //Занесение полученный данных в структуры главной формы
                    FillMainStruct(index, i);
                }
                //Красим батон
                button5.BackColor = Color.GreenYellow;
                button4.BackColor = Color.GreenYellow;
                button3.BackColor = Color.GreenYellow;
            }
        }

        private void button8_Click(object sender, EventArgs e)
        {
            FormMemory fm = new FormMemory(mainForm);
            fm.Show();
        }
    }
}
```

Файл *FormMU.cs*:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace model1
{
    public partial class FormMU : Form
    {
        Form1 mainForm;

        public FormMU(Form1 form)
        {
            InitializeComponent();
            this.mainForm = form;
            textBox11.Text = "1";
            FillForm(1);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            bool err = false;
            err = CheckForm();
            if (!err)
            {
                //Занесение полученный данных в структуры главной формы
                FillMainStruct(Convert.ToInt32(textBox11.Text));
                //Закрываем форму
               // this.Close();
               //Отображаем на форме
                label16.Text     =     String.Format("Применены     настройки     к     УУ     №{0}",
Convert.ToInt32(textBox11.Text));
            }
        }

        public bool CheckForm()
        {
            bool err = false;
            if (textBox1.Text != "" || (textBox6.Text != "" && textBox7.Text != ""))
            {
                if (radioButton1.Checked)
                {
                    try
                    {
                        if (Convert.ToInt32(textBox1.Text) < 0)
                        {
                            MessageBox.Show("Мат. ожидание меньше нуля!");
                            err = true;
                        }
                    }
                    catch (FormatException)
                    {
                        MessageBox.Show("Введенное значение мат. ожидания имеет неверный формат!");
                        err = true;
                    }
                }
                else if (radioButton2.Checked)
                {
                    try
                    {
                        if (Convert.ToInt32(textBox6.Text) < 0 || Convert.ToInt32(textBox7.Text) < 0)
                        {
                            MessageBox.Show("Задержка должна быть больше нуля!");
                            err = true;
                        }
```

```
                        if (Convert.ToInt32(textBox6.Text) > Convert.ToInt32(textBox7.Text))
                        {
                            MessageBox.Show("Минимальное значение задержки больше, чем максимальное!");
                            err = true;
                        }
                    }
                    catch (FormatException)
                    {
                        MessageBox.Show("Введенное значение задержки имеет неверный формат!");
                        err = true;
                    }
                }
                else if (radioButton3.Checked)
                {
                    if (Convert.ToInt32(textBox17.Text) < 0)
                    {
                        MessageBox.Show("Период выдачи не может быть меньше нуля!\n Текущее занчение: "
+ textBox17.Text);
                    }
                }
                if    ((Convert.ToInt32(textBox12.Text)    +    Convert.ToInt32(textBox13.Text)    +
Convert.ToInt32(textBox14.Text)  +  Convert.ToInt32(textBox4.Text)  +  Convert.ToInt32(textBox3.Text)  +
Convert.ToInt32(textBox5.Text)  +  Convert.ToInt32(textBox10.Text)  +  Convert.ToInt32(textBox8.Text)  +
Convert.ToInt32(textBox9.Text)) != 100)
                {
                    MessageBox.Show("Вероятности появления команд различного типа в\r\n сумме должны
давать 100% !");
                    err = true;
                }
                if (Convert.ToInt32(textBox16.Text) < 0 || Convert.ToInt32(textBox16.Text) > 100)
                {
                    MessageBox.Show("Не корректное значение вероятности условного перехода!");
                    err = true;
                }
                if (Convert.ToInt32(textBox15.Text) < 0 || Convert.ToInt32(textBox15.Text) > 100)
                {
                    MessageBox.Show("Не корректное значение вероятности блока предсказания!");
                    err = true;
                }
                try
                {
                    if (Convert.ToInt32(textBox18.Text) < 0)
                    {
                        MessageBox.Show("Мат. ожидание меньше нуля!");
                        err = true;
                    }
                }
                catch (FormatException)
                {
                    MessageBox.Show("Введенное значение мат. ожидания имеет неверный формат!");
                    err = true;
                }
            }
            else
            {
                MessageBox.Show("Не все параметры указаны!");
                err = true;
            }
            return err;
        }

        protected void FillForm(int index)
        {
            label16.Text = "";
            index--;
            textBox12.Text = mainForm.mu_params.MUList[index].FUIntASExp.ToString();
            textBox13.Text = mainForm.mu_params.MUList[index].FUIntMExp.ToString();
            textBox14.Text = mainForm.mu_params.MUList[index].FUIntUExp.ToString();
            textBox10.Text = mainForm.mu_params.MUList[index].FUFloatASExp.ToString();
            textBox9.Text = mainForm.mu_params.MUList[index].FUFloatMExp.ToString();
            textBox8.Text = mainForm.mu_params.MUList[index].FUFloatDExp.ToString();
            textBox4.Text = mainForm.mu_params.MUList[index].FUMemoryR1Exp.ToString();
            textBox3.Text = mainForm.mu_params.MUList[index].FUMemoryR2Exp.ToString();
            textBox5.Text = mainForm.mu_params.MUList[index].FUMemoryWExp.ToString();
            textBox18.Text = mainForm.mu_params.MUList[index].RefreshMathExp.ToString();
            radioButton1.Checked = mainForm.mu_params.MUList[index].law == 1 ? true : false;
            radioButton2.Checked = mainForm.mu_params.MUList[index].law == 2 ? true : false;
```

```
            radioButton3.Checked = mainForm.mu_params.MUList[index].law == 3 ? true : false;
            textBox2.Text = mainForm.mu_params.MUCount.ToString();
            textBox16.Text = mainForm.mu_params.MUList[index].ExpBranch.ToString();
            textBox15.Text = mainForm.mu_params.MUList[index].ExpPB.ToString();
            if (radioButton1.Checked)
            {
                textBox1.Text = mainForm.mu_params.MUList[index].MathExp.ToString();
            }
            else if (radioButton2.Checked)
            {
                textBox6.Text = mainForm.mu_params.MUList[index].FUMin.ToString();
                textBox7.Text = mainForm.mu_params.MUList[index].FUMax.ToString();
            }
            else if (radioButton3.Checked)
            {
                textBox17.Text = mainForm.mu_params.MUList[index].Period.ToString();
            }
        }

        private void FillMainStruct(int index)
        {
            index--;
            mainForm.mu_params.MUList[index].FUIntASExp = Convert.ToInt32(textBox12.Text);
            mainForm.mu_params.MUList[index].FUIntMExp = Convert.ToInt32(textBox13.Text);
            mainForm.mu_params.MUList[index].FUIntUExp = Convert.ToInt32(textBox14.Text);
            mainForm.mu_params.MUList[index].FUFloatASExp = Convert.ToInt32(textBox10.Text);
            mainForm.mu_params.MUList[index].FUFloatMExp = Convert.ToInt32(textBox9.Text);
            mainForm.mu_params.MUList[index].FUFloatDExp = Convert.ToInt32(textBox8.Text);
            mainForm.mu_params.MUList[index].FUMemoryR1Exp = Convert.ToInt32(textBox4.Text);
            mainForm.mu_params.MUList[index].FUMemoryR2Exp = Convert.ToInt32(textBox3.Text);
            mainForm.mu_params.MUList[index].FUMemoryWExp = Convert.ToInt32(textBox5.Text);
            mainForm.mu_params.MUList[index].RefreshMathExp = Convert.ToInt32(textBox18.Text);
            if(radioButton1.Checked)
            {
                mainForm.mu_params.MUList[index].law = 1;
            }
            else if (radioButton2.Checked)
            {
                mainForm.mu_params.MUList[index].law = 2;
            }
            else if (radioButton1.Checked)
            {
                mainForm.mu_params.MUList[index].law = 3;
            }
            mainForm.mu_params.MUCount = Convert.ToInt32(textBox2.Text);
            mainForm.mu_params.MUList[index].ExpBranch = Convert.ToInt32(textBox16.Text);
            mainForm.mu_params.MUList[index].ExpPB = Convert.ToInt32(textBox15.Text);
            if (radioButton1.Checked)
            {
                mainForm.mu_params.MUList[index].MathExp = Convert.ToInt32(textBox1.Text);
            }
            else if (radioButton2.Checked)
            {
                mainForm.mu_params.MUList[index].FUMin = Convert.ToInt32(textBox6.Text);
                mainForm.mu_params.MUList[index].FUMax = Convert.ToInt32(textBox7.Text);
            }
            else if (radioButton3.Checked)
            {
                mainForm.mu_params.MUList[index].Period = Convert.ToInt32(textBox17.Text);
            }
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            //Показательный
            label2.Enabled = true;
            textBox1.Enabled = true;
            textBox1.Text         =         mainForm.mu_params.MUList[Convert.ToInt32(textBox11.Text)-
1].MathExp.ToString();

            textBox17.Enabled = false;
            label19.Enabled = false;
            label11.Enabled = false;
            label3.Enabled = false;
            label4.Enabled = false;
            textBox6.Enabled = false;
            textBox7.Enabled = false;
```

```
 //           textBox6.Text = "";
 //           textBox7.Text = "";
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            //Равномерный
            label11.Enabled = true;
            label3.Enabled = true;
            label4.Enabled = true;
            textBox6.Enabled = true;
            textBox6.Text         =         mainForm.mu_params.MUList[Convert.ToInt32(textBox11.Text)-
1].FUMin.ToString();
            textBox7.Enabled = true;
            textBox7.Text         =         mainForm.mu_params.MUList[Convert.ToInt32(textBox11.Text)-
1].FUMax.ToString();

            label2.Enabled = false;
            textBox1.Enabled = false;
            textBox17.Enabled = false;
            label19.Enabled = false;
//            textBox1.Text = "";
        }

        private void textBox11_TextChanged(object sender, EventArgs e)
        {
            if      (Convert.ToInt32(textBox11.Text)      <=      mainForm.mu_params.MUCount      &&
Convert.ToInt32(textBox11.Text) >= 1)
            {
                try
                {
                    FillForm(Convert.ToInt32(textBox11.Text));
                }
                catch
                { }
            }
            else
            {
                MessageBox.Show(String.Format("Некорректный   индекс   блока   УУ!\n   Текущий:   {0}",
Convert.ToInt32(textBox11.Text)));
                textBox11.Text = "1";
                FillForm(1);
            }
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            mainForm.mu_params.MUCount = Convert.ToInt32(textBox2.Text);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            textBox12.Text = "0";
            textBox13.Text = "0";
            textBox14.Text = "0";
            textBox10.Text = "0";
            textBox9.Text = "0";
            textBox8.Text = "0";
            textBox4.Text = "0";
            textBox3.Text = "0";
            textBox5.Text = "0";
            label16.Text = String.Format("Значения вероятностей обнулены");
        }

        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            //Детерминированный
            textBox17.Enabled = true;
            textBox17.Text         =         mainForm.mu_params.MUList[Convert.ToInt32(textBox11.Text)-
1].Period.ToString();
            label19.Enabled = true;

            label2.Enabled = false;
            textBox1.Enabled = false;
            label11.Enabled = false;
            label3.Enabled = false;
            label4.Enabled = false;
```

```
            textBox6.Enabled = false;
            textBox7.Enabled = false;
//            textBox6.Text = "";
//            textBox7.Text = "";
//            textBox1.Text = "";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            bool err = false;
            err = CheckForm();
            if (!err)
            {
                for (int i = 0; i < mainForm.mu_params.MUCount; i++)
                {
                    //Занесение полученный данных в структуры главной формы
                    FillMainStruct(Convert.ToInt32(i));
                }
                //Закрываем форму
                this.Close();
            }
        }
    }
}
```

ДОДАТОК В

Компьютерна презентація

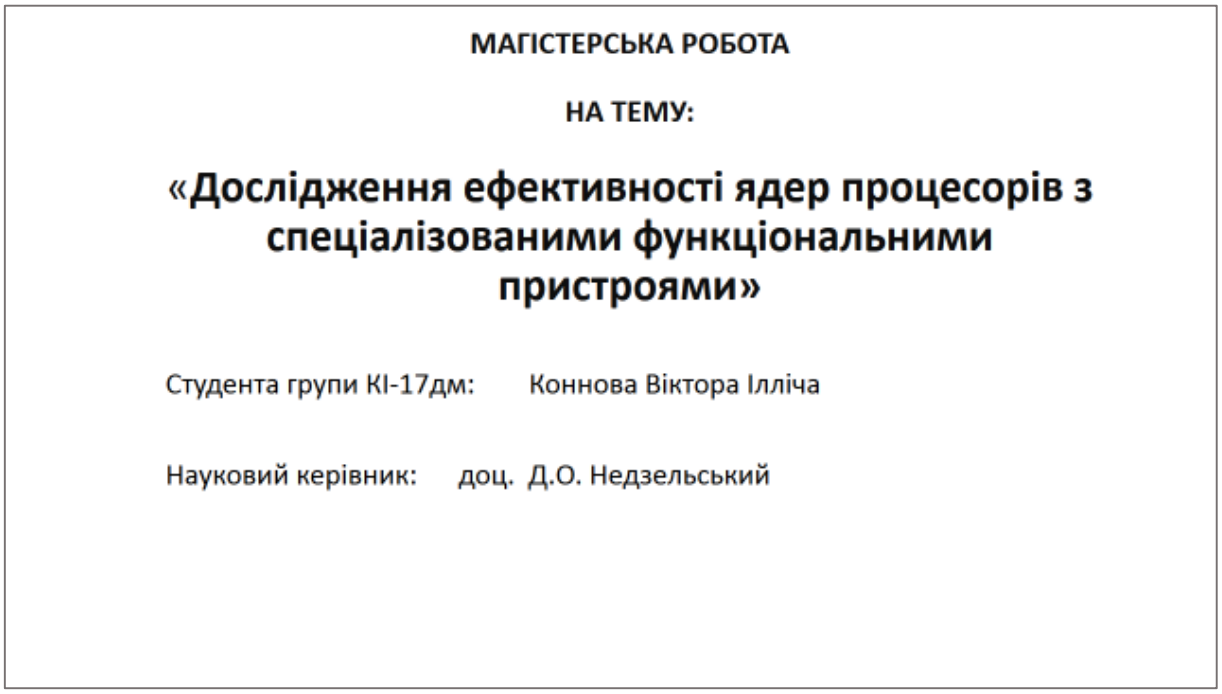


Рисунок В.1 - Презентація. Титульний аркуш

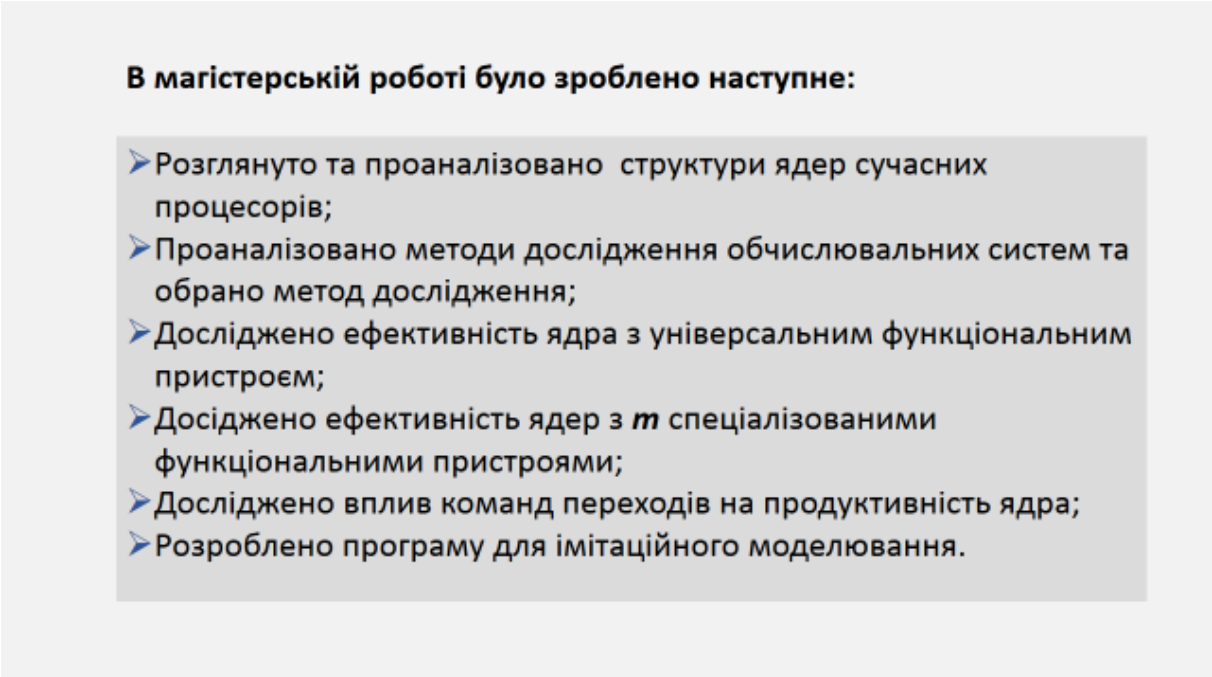


Рисунок В.2 - Презентація. Перелік задач

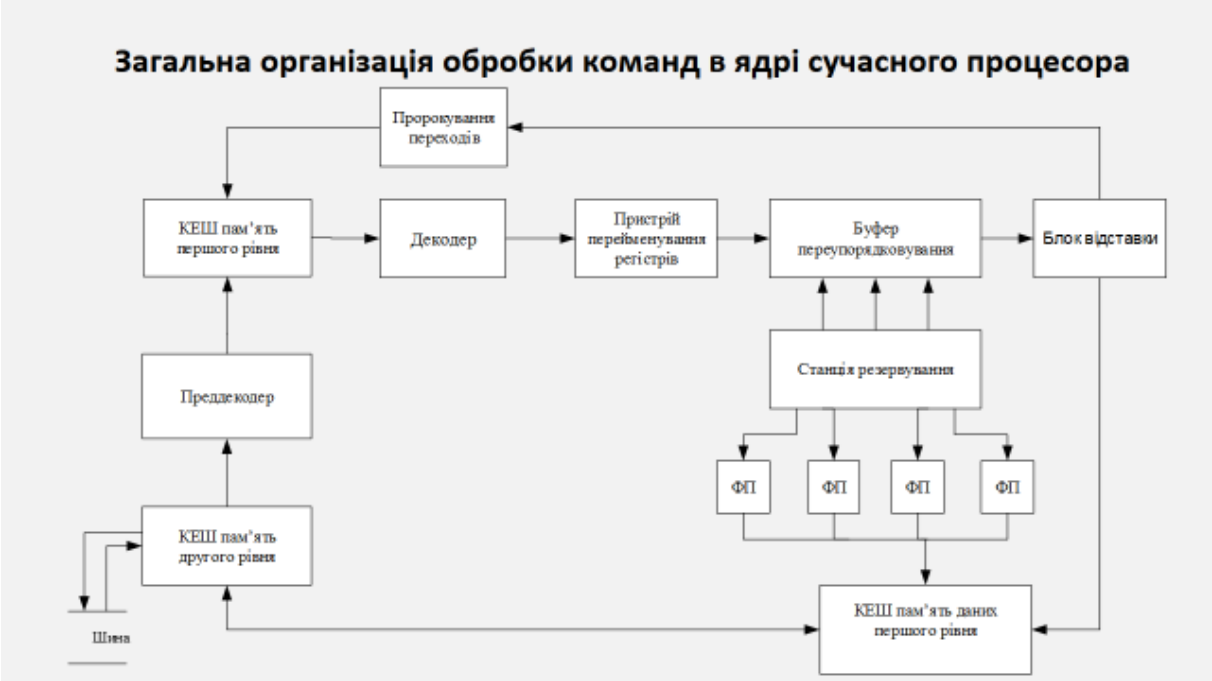


Рисунок В.3 - Презентація. Загальна організація обробки команд в ядрі сучасного процесора

Методи дослідження обчислювальних систем:

1. Аналітичні. Аналітичні методи дослідження систем зводяться до побудови математичних моделей, які представляють фізичні властивості як математичні об'єкти й відносини між ними, які виражаються за допомогою математичних операцій. Моделі, побудовані цими методами, називаються аналітичними моделями.

2. Чисельні. Чисельні методи ґрунтуються на побудові кінцевої послідовності дій над числами, що приводить до отримання необхідних результатів. При наявності математичної моделі досліджуваного об'єкта застосування чисельних методів зводиться до заміни математичних операцій і відносин відповідними операціями над числами: заміні інтегралів сумами, похідних різницевим відносинами, нескінченних сум кінцевими і т. д. Результат застосування чисельних методів - таблиці (графіки) залежностей, які розкривають властивості об'єкта.

3. Імітаційні. Імітаційні методи засновані на поданні порядку функціонування системи у вигляді алгоритму, який називається імітаційною (алгоритмічною) моделлю. Програма містить процедури, які реєструють стани імітаційної моделі й обробляють зареєстровані дані для оцінки необхідних характеристик процесів і системи, що моделюється.

4. Експериментальні. Експериментальні методи ґрунтуються на одержанні даних про функціонування існуючих систем у реальних або спеціально створених умовах з метою оцінки якості функціонування й виявлення залежностей, що характеризують властивості систем і їх складових. Типові задачі, котрі розв'язуються експериментальними методами, - оцінка продуктивності й надійності системи, визначення складу й кількісних показників системного навантаження залежно від прикладного навантаження і т.п.

Рисунок В.4 - Презентація. Методи дослідження обчислювальних систем

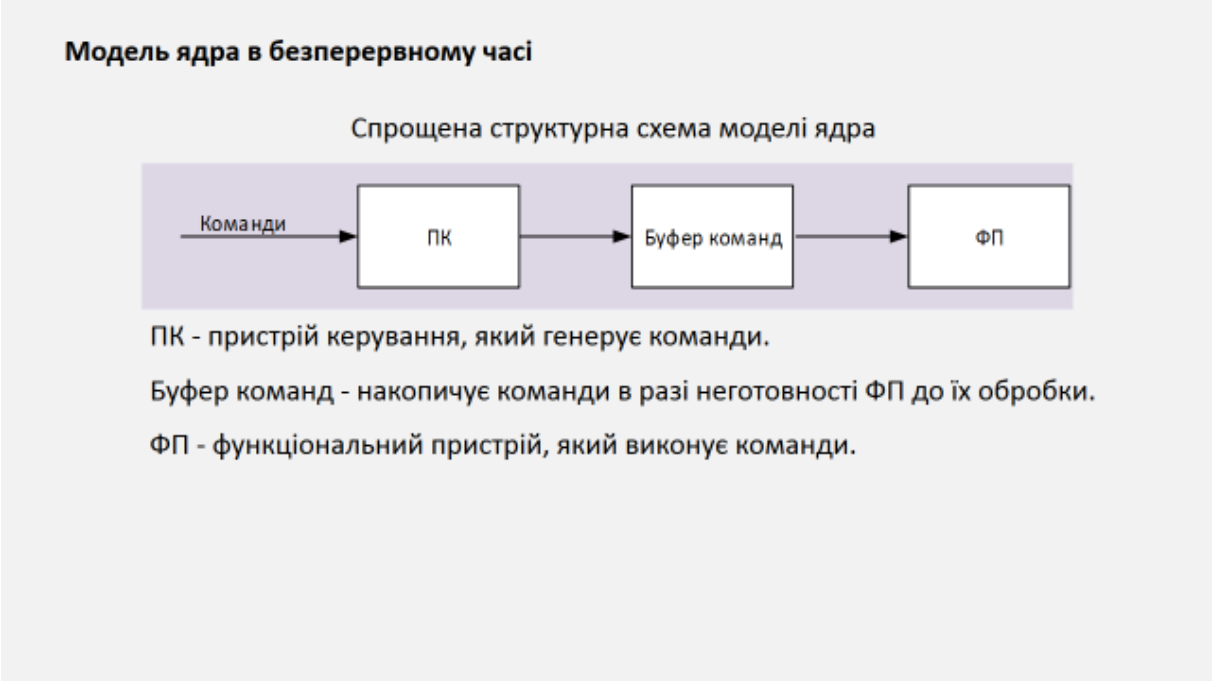


Рисунок В.5 – Презентація. Модель ядра в безперервному часі

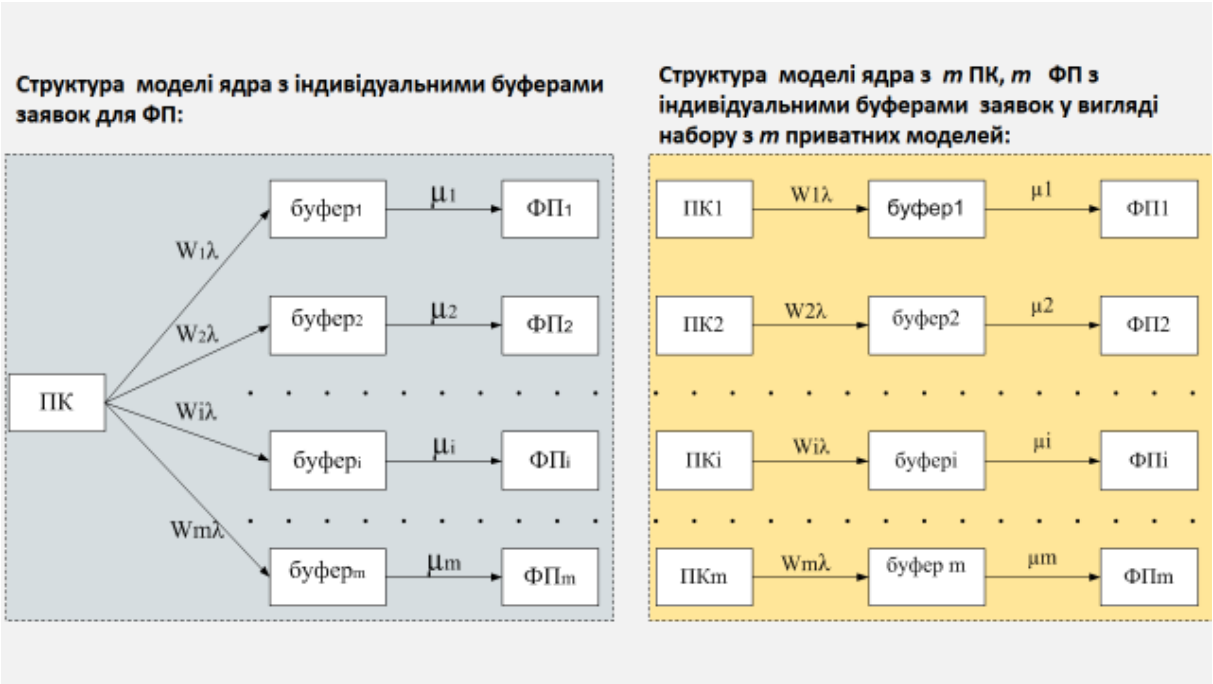


Рисунок В.6 – Презентація. Структури моделей ядер

Коефіцієнт використання ФП

$$H_{ФП} = \frac{\rho(1-\rho^{n+1})}{1-\rho^{n+2}} \quad \text{для} \quad \rho \neq 1$$

$$H_{ФП} = \frac{n+1}{n+2} \quad \text{для} \quad \rho = 1$$

Коефіцієнт використання ПК

$$E_{ПК} = \frac{1-\rho^{n+1}}{1-\rho^{n+2}} \quad \text{для} \quad \rho \neq 1$$

$$E_{ПК} = \frac{n+1}{n+2} \quad \text{для} \quad \rho = 1$$

Рисунок В.7 – Презентація. Коефіцієнти використання ФП і ПК

Показники ефективності (значення коефіцієнтів використання пристроїв) як функції розміру буфера при ρ = 1

n		2	4	8	16	32	64
H	Аналітичні результати	0.750	0.833	0.900	0.944	0.971	0.985
E		0.750	0.833	0.900	0.944	0.971	0.985
H	Чисельний експеримент	0.75	0.834	0.9	0.945	0.970	0.988
E		0.75	0.834	0.9	0.944	0.971	0.985

Рисунок В.8 – Презентація. Показники ефективності як функції розміру буфера при ρ=1

Показники ефективності (значення коефіцієнтів використання пристроїв) як функції розміру буфера при ρ >1 (ρ=2)

n		2	4	8	16	32	64
H	Аналітичні результати	0,857	0,933	0,984	0,999	1	1
E		0,429	0,467	0,492	0.4995	0,5	0,5
H	Чисельний експеримент	0,938	0,975	0,995	1	1	1
E		0,429	0,467	0,492	0,499	0,5	0,5

Рисунок В.9 – Презентація. Показники ефективності як функції розміру буфера при ρ=2

Показники ефективності (значення коефіцієнтів використання пристроїв) як функції розміру буфера при ρ <1 (ρ=0,5)

n		2	4	8	16	32	64
H	Аналітичні результати	0,429	0,467	0,492	0,4995	0,5	0,5
E		0,857	0,933	0,984	0,9990	1	1
H	Чисельний експеримент	0,429	0,467	0,492	0,4995	0,541	0,556
E		0,857	0,933	0,984	0,999	1	1

Рисунок В.10 – Презентація. Показники ефективності як функції розміру буфера при ρ=0,5

Результати моделей з 2 ФП при ρ_1= 0,5; ρ_2=0,6

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,731	0,851	0,364	0,365	0,433	0,438
2	0,853	0,914	0,429	0,427	0,506	0,512
4	0,954	0,969	0,481	0,477	0,568	0,572
8	0,996	0,996	0,502	0,498	0,596	0,597
16	1,000	1,000	0,505	0,500	0,597	0,600
32	1,000	1,000	0,505	0,500	0,597	0,600

Рисунок В.11 – Презентація. Результати моделей з 2 ФП при ρ_1=0,5; ρ_2=0,6

Результати моделей з 2 ФП при ρ_1= 1,2; ρ_2=0,8

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,492	0,597	0,727	0,731	0,291	0,316
2	0,562	0,629	0,855	0,849	0,335	0,351
4	0,627	0,652	0,946	0,946	0,377	0,381
8	0,660	0,663	0,992	0,991	0,392	0,396
16	0,667	0,666	1,000	1,000	0,400	0,400
32	0.667	0.666	1,000	1,000	0,400	0,400

Рисунок В.12 – Презентація. Результати моделей з 2 ФП при ρ_1=1,2; ρ_2=0,8

Результати моделей з 2 ФП при ρ_1= 1,5; ρ_2=0,6

n	$E_{ПК}$		$H_{ФП1}$		$H_{ФП2}$	
	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель	Еталонна модель	Спрощена модель
1	0,731	0,851	0,364	0,365	0,433	0,438
2	0,853	0,914	0,429	0,427	0,506	0,512
4	0,954	0,969	0,481	0,477	0,568	0,572
8	0,996	0,996	0,502	0,498	0,596	0,597
16	1,000	1,000	0,505	0,500	0,597	0,600
32	1,000	1,000	0,505	0,500	0,597	0,600

Рисунок В.13 – Презентація. Результати моделей з 2 ФП при ρ_1=1,5; ρ_2=0,6

Головне вікно програми імітаційної моделі системи з індивідуальними буферами (без урахування "перешкод" роботі конвеєра)

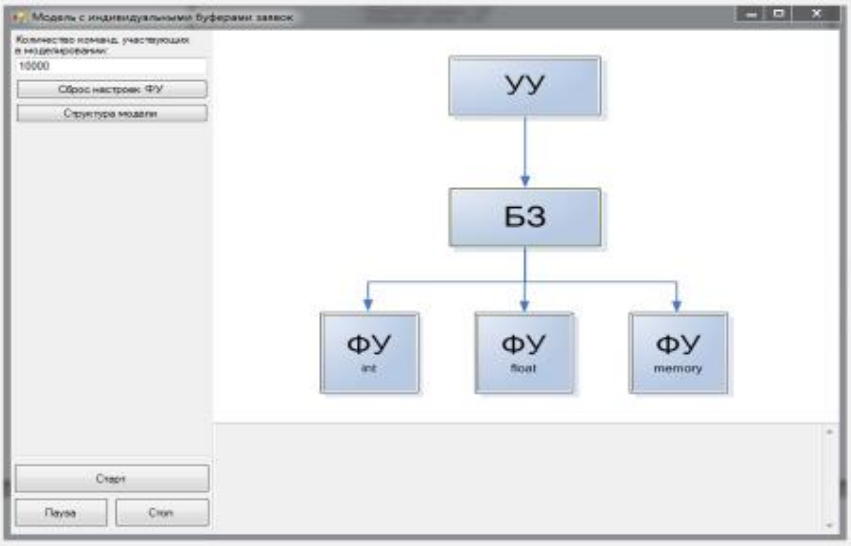


Рисунок В.14 – Презентація. Головне вікно програми імітційної моделі системи з індивідуальними буферами

Програма розрахунку коефіцієнтів використання системи з урахуванням умовних переходів (з блоком передбачення переходів):

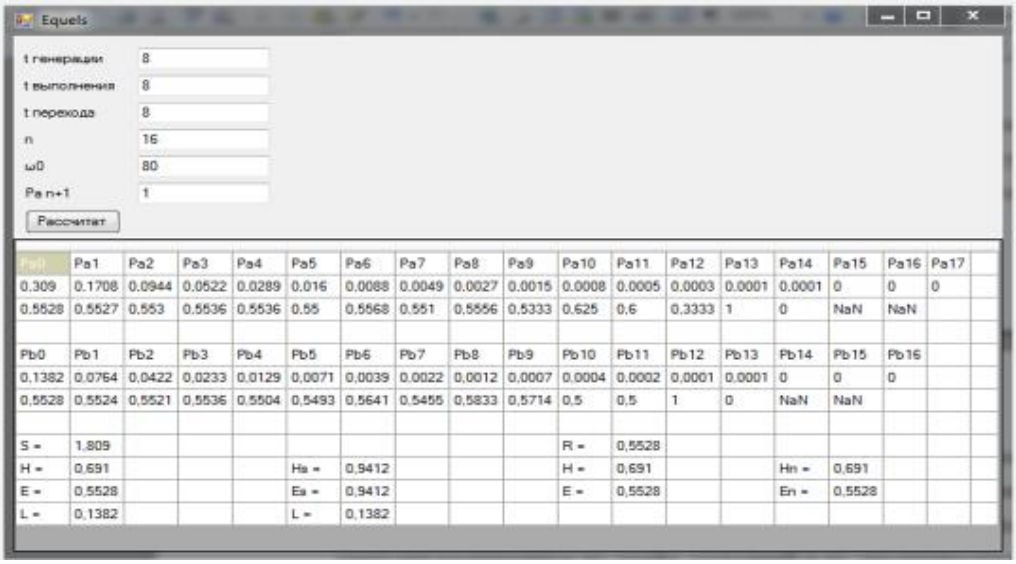


Рисунок В.15 – Презентація. Головне вікно програми розрахунку коефіцієнтів використання системи з урахуванням умовних переходів

Висновки

В магістерській роботі:

В розділі 1 розглянуті структури ядер сучасних процесорів на прикладі структури Haswel фірми Intel, їх основні блоки, організация та взаємодія. Розглянуті основні методи моделювання та проведено вибір прийнятних з них для створення моделей ядер процесорів.

В розділі 2 розроблені аналітичні модели конвеєрного ядра процесора з універсальним функціональним пристроєм. Проведені дослідження системи з универсальним функціональним пристроєм. Розроблені аналітичні моделі конвеєрного ядра з m спеціалізованими функціональними пристроями. Проведені дослідження створених моделей.

В розділі 3 розроблена аналітична модель ядра процесора з врахуванням впливу «перешкод» роботі конвеєра без блоку передбачення адрес переходів. Користуючись створеною моделлю проведено дослідження.

В ході практичної частини роботи були отримані наступні результати:

1) Розроблено аналітичні та імітаційні моделі конвеєрних ядер з універсальними функціональними пристроями.
2) Розроблено аналітичні та імітаційні моделі конвеєрних ядер з будь-якою кількістю функціональних пристроїв.
3) Розроблено аналітичні та чисельні моделі конвеєрних ядер з урахуванням "перешкод" роботі конвеєра.

Рисунок В.15 – Презентація. Висновки