Multiscale model simul. – 2005. – Vol. 4, № 2. – P. 490–530. 16. Dabov K., Foi A., Katkovnik V., Egiazarian K. Image denoising by sparse 3D transform-domain collaborative filtering // IEEE Trans. Image Processing. – 2007. – Vol. 16, № 8.

УДК 004.31, 004.23

Г.В. Кицун
Національний університет "Львівська політехніка",
кафедра електронних обчислюваних машин

ВИБІР МЕТОДІВ АДРЕСАЦІЇ ПАМ'ЯТІ УНІВЕРСАЛЬНОГО КОМП'ЮТЕРА

Розглядаються методи адресації універсальних комп'ютерів архітектур CISC, RISC та проводиться їх аналіз. На основі існуючих методів розробляються методи адресації пам'яті універсального комп'ютера, який проектується сьогодні.

The universal CISC, RISC computer architectures modes of adressing are examined and their analysis is conducted in this article. On the basis of existent methods, the methods of adressing for the perfected universal computer, which is designed presently are developed.

Вступ. Методи адресації пам'яті – це комплекс стандартизованих для певної архітектури системи команд комп'ютера способів для визначення (обчислення) місця розташування операндів в пам'яті комп'ютера або адреси наступної команди при виконанні команд переходу [10].

Для того, щоб отримати можливість використовувати дані з пам'яті в обчислювальних операціях, необхідно однозначно вказати процесору їхнє місцезнаходження. В фон-нейманівських машинах кожна комірка пам'яті має власну адресу й проблема визначення місця розташування потрібних даних зводиться до визначення цієї адреси. В перших комп'ютерах адресу або номер комірки необхідно було вказувати явно, і такий метод адресації виявився дуже незручним. Труднощі в алгоритмізації різних задач, де була потрібна автоматизація процесу визначення адреси, стали причиною введення згодом широкого спектра методів адресації. Кожний з них фактично пропонує певну формулу для обчислення ефективної (тобто фактичної) адреси, зручну в тій або іншій ситуації. Пік винахідництва у цій галузі припав на час панування архітектур типу CISC [5], які давали змогу безпосередньо використовувати як один з операндів комірку пам'яті. Архітектури RISC [12] типу "регістр-регістр", в яких доступ до пам'яті регламентується значно більш жорстко, мають, у порівнянні з CISC, дуже скромний набір методів адресації.

Постановка задачі. Проаналізувати існуючі методи адресації та запропонувати методи адресації пам'яті універсального комп'ютера, які б давали змогу спростити декодування команд.

Методи адресації пам'яті в комп'ютерах з архітектурою CISC. Як приклад CISC-моделі адресації пам'яті розглянемо різноманітну палітру методів популярної колись архітектури VAX-11 на прикладі двоопернадної команди додавання ADD a,b (a:=a+b), яка зображена на рис. 1, де кожний операнд може бути як в регістрі, так і в пам'яті. Поля "режим" визначають режим адресації, поля "регістр" – номери задіяних регістрів (рис. 1).

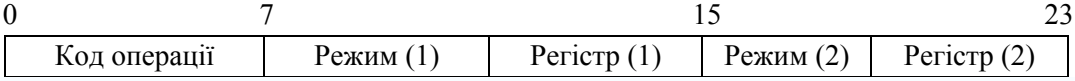

Рис. 1. Формат двоопернадної команди архітектури VAX [6]

У табл. 1 [6] зведені основні методи адресації VAX (загальна їх кількість перевищує два десятки, але інші є похідними від наведених) за умови, що операнд a завжди перебуває в регістрі R1, а ефективна адреса операнда b обчислюється залежно від зазначеного в команді методу адресації.

Таблиця 1

Основні методи адресації VAX

Способ адресації	Запис команди	Ефективна адреса	Розгорнутий запис	Застосування
Абсолютна (пряма)	ADD R1,@#1000	M[1000]	R1:=R1+M[1000]	Коли відома абсолютна (пряма) адреса операнда
Безпосередня	ADD R1,#4	-	R1:=R1+4	Один з операндів – константа (арифм. операції, перевірки умов)
Регістрова	ADD R1,R2	R2	R1:=R1+R2	Усі операнди в регістрах
Непряма регістрова	ADD R1,(R2)	M[R2]	R1:=R1+M[R2]	Доступ до даних за попередньо обчисленою адресою, визначення адреси, на яку посилається вказівник (адреса вказівника – в R2)
Непряма	ADD R1,@(R2)	M[M[R2]]	R1:=R1+M[M[R2]]	Робота з вказівниками: якщо в R2 – адреса вказівника p, то ефективна адреса – це значення *p
За зсувом (базова, індексна)	ADD R1,30(R2)	M[R2+30]	R1:=R1+M[R2+30]	Один з основних способів. Застосовується для організації переміщуваних програм (фіксація «початку відліку (база)» в R2), для роботи з масивами (адреса початку – в R2, змінною зсуву отримаємо доступ до різних комірок масиву)
Маштабування (індексна регістрова непряма)	ADD R1,(R2)[R3]	M[x*R3+R2]	R1:=R1+M[x*R3+R2] x – різне в залежності від типу операндів	Робота з масивами
Непряма регістрова з автоінкрементом	ADD R1,(R2)+	M[R2]	R1:=R1+M[R2]; R2:=R2+1	Робота с масивами в циклах. R2 початково вказує на початок масиву, кожна нова ітерація супроводжується позиціонуванням на наступний елемент
Непряма регістрова з автодекрементом	ADD R1,-(R2)	M[R2-1]	R2:=R2-1; R1:=R1+M[R2]	Аналогічно попередньому способу

Примітки: Ri – регістр з порядковим номером i. M[j] – комірка пам’яті з абсолютною адресою j. M[Ri]– комірка пам’яті з адресою, яка знаходиться в регістрі Ri

Інформація про операнд міститься в його специфікаторі, формат якого варіюється, але для більшості методів адресації складається з полів “регістр” і “режим”. Тобто, режим адресації задається полем “режим”, а номер задіяного регістра втримується в полі “регістр”. У режимі адресації “за зсувом” (базова, індексна), в команді після полів режиму й регістра з’являється додаткове поле із зазначенням величини самого зсуву. В режимі масштабування також застосовується розширення команди й у додатковому полі знаходиться специфікатор для третього задіяного регістра. Абсолютна адресація реалізується через непряму регістрову з автоінкрементом, а безпосередня – через непряму з автоінкрементом та дописуванням зсуву або константи безпосередньо після команди. При цьому регістр PC у момент виконання команди завжди вказує на наступну після неї комірку пам’яті, яка містить адресу зсуву або константи, й обчислення абсолютної адреси відбудеться коректно. Після цього необхідно лише збільшити PC до адреси наступної команди. Це робиться шляхом додавання до поточного значення лічильника одиниці (автоінкремента).

Методи адресації пам’яті в комп’ютерах з архітектурою RISC. В архітектурах RISC (регістр–регістр) про методи адресації пам’яті має сенс говорити тільки щодо спеціальних команд завантаження й збереження даних. Всі обчислювальні команди в цих архітектурах використовують винятково регістри або константи в полі команди, а отже тільки регістрову або безпосередню адресацію. Архітектури RISC, які ми маємо на увазі, говорячи про машини типу «регістр–регістр» (хоча, якщо точніше, RISC є всього лише підмножиною цього класу), власне, використовують дуже обмежений набір методів адресації, які несуть інше функціональне навантаження й не є складовою частиною обчислювальних операцій.

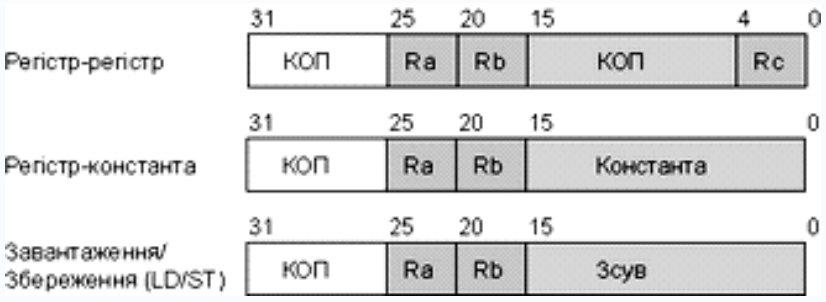

Рис. 2. Формати команд архітектури DEC Alpha [6]

Під ефективною адресою тепер розумітимемо адресу комірки пам’яті, призначеної для завантаження в регістр спеціальною командою LD (load) або збереження результату спеціальною командою ST (store) [6].

Цей набір є досить скромним порівняно з різноманіттям VAX. Разом з нею, архітектура DEC Alpha з її єдиним методом адресації, виглядає, на перший погляд, навіть дивно, тим більше, що довгий час процесорам саме цієї архітектури належав світовий рекорд швидкодії. Однак, як показують статистичні дослідження, таке рішення цілком обґрунтоване: найуживанішими є методи адресації за зсувом (базової) й з використанням константи в полі команди, на них припадає в середньому стільки ж, скільки й на всі інші разом взяті. Хоча, звичайно ж, дуже багато залежить від стратегії оптимізації, використовуваної компілятором для конкретної архітектури [6]. У табл. 2 показано, які з способів адресації використовуються в сучасних процесорах.

Точно так, як в VAX, деякі методи адресації були спеціальними випадками інших, так і в Alpha на основі адресації за зсувом (базової) можна отримати, наприклад, регістрову адресацію зазначенням нуля в якості розміру зсуву, або абсолютну адресацію записом нуля в якості задіяного регістра (зазвичай замість цього використовується регістр R0, який в архітектурах RISC містить константу нуля).

Таблиця 2

Способи адресації в сучасних процесорах

Способ адресації	Запис команди	Ефективна адреса	Розгорнутий запис	MIPS IV	PA-RISC 2.0	PowerPC	DEC Alpha	SPARC v9
За зсувом	LD R1,30(R2) ST R1,30(R2)	M[30+R2]	R1:=M[30+R2] R1:=M[30+R2]	так	так	так	так	ні
Регістрово-індексна	LDX R1,R2,R3 STX R1,R2,R3	M[R2+R3]	R1:=M[R2+R3] R1:=M[R2+R3]	частково	частково	так	ні	ні
Регістрова з поновленням регістра	LDU R1,50(R2) STU R1,50(R2)	M[50+R2]	R1:=M[30+R2]; R2:=50+R2 R1:=M[30+R2]; R2:=50+R2	ні	так	так	ні	ні
Регістрово-індексна з поновленням регістра	LDUX R1,R2,R3 STUX R1,R2,R3	M[R2+R3]	R1:=M[R2+R3]; R2:=R2+R3 R1:=M[R2+R3]; R2:=R2+R3	ні	так	так	ні	ні
Регістрова (похідна від адресації за зсувом)	LD R1,0(R2)	R2	R1:=R2	так	так	так	так	так
Абсолютна (похідна від адресації за зсувом)	LD R1,500(R0)	M[500]	R1:=M[500]	так	так	так	так	так

Порівняльний аналіз та вибір методів адресації пам’яті. З розглянутих вище архітектур можна виділити такі недоліки:

- в архітектурі CISC багато різних методів адресації, які дуже рідко використовуються, що ускладнює апаратуру і тим самим накладає певні обмеження на швидкодію;
- в архітектурі CISC для визначення режимів адресації відводяться додаткові поля в інструкції для кожного з операндів;
- в архітектурі CISC на виконання команди відводиться декілька тактів залежно від складності методу адресації, що ускладнює проходження команди по конвеєру і створює різного роду затримки в конвеєрі;
- в архітектурі RISC завдяки вказанню двох адрес операндів та адреси призначення обчисленої команди зменшується кількість регістрів загального призначення;
- в архітектурі RISC частина розрядів в коротких командах не використовується;
- в архітектурі RISC при виконанні операцій “регістр–пам'ять” довге поле адреси пам’яті в інструкції скорочує місце під номер регістра, що обмежує загальну кількість регістрів загального призначення (РЗП);
- в обох архітектурах присутня прив’язка до адресації регістрів в регістровій пам’яті процесора;
- в обох архітектурах в регістровій пам’яті процесора присутні регістри спеціального призначення, що вводить деяке обмеження щодо універсальності даних архітектур.

У регістровій пам’яті універсального комп’ютера можуть зберігатись константи, результати обчислень, адреси комірок пам’яті, вміст елементів пам’яті тощо. Для звернення до елемента пам’яті процесор обчислює ефективну адресу пам’яті.

За результатами аналізу можна запропонувати для реалізації такі методи адресації:

- базову із зсувом, безпосередню, регістрову, оскільки вони є найпоширенішими [1–5, 8–12]. Основне питання, яке виникає для методу базової адресації із зсувом, пов’язане з довжиною (розрядністю) зсуву. Вибір довжини зсуву визначає довжину команди. Безпосередню адресацію використовують при виконанні арифметичних операцій, операцій порівняння, а також для завантаження констант в регістри;
- непряму регістрову адресацію, яка дає змогу звертатись до комірок пам’яті за вмістом регістра, що часто використовується при зверненні до пам’яті за покажчиком, або попередньо визначеною адресою;
- регістрово-індексну адресацію з поновленням регістра, яка дає змогу під час роботи з масивами в одному регістрі зберігати базу масиву, а в іншому – індекс масиву, а також результат, який утворився в процесі додавання бази та індексу;
- пряму (абсолютну) адресацію, яка дає змогу звертатись до пам’яті безпосередньо за допомогою вказаної константи, що вказує на комірку пам’яті, в якій (переважно) знаходяться статичні дані.

В існуючих процесорах адресація регістрів та методи адресації не пов’язані з лічильником команд. Такий вид адресації використовується переважно для визначення програмних адрес в командах передачі управління. У запропонованих методах адресації пам’яті універсального комп’ютера адресація регістрів та лічильник команд (програмний лічильник) тісно пов’язані. Адресація регістрів напряму залежить від програмного лічильника, а програмний лічильник, своєю чергою, залежить від команд управління. Значення програмного лічильника є адресою для регістрової пам’яті, структура якої передбачає розміщення двох операндів за однією адресою, що дає змогу спростити опис команди, зменшивши поле для вказання адреси отримуваних операндів та, відповідно, спростити процес декодування команд.

У табл. 3 на прикладі команди завантаження (Load) наведено запропоновані методи адресації пам’яті універсального комп’ютера. У цій таблиці знак “(” використовується для позначення оператора присвоєння, буква M позначає пам’ять (Memory), RM позначає регістрову пам’ять (Register Memory), а RTC_1 (Real-Time Clock) позначає першу половину комірки регістрової пам’яті з номером, який відповідає покажчику програмного лічильника процесора. Отже, M[RTC_1] позначає вміст комірки пам’яті, адреса якої визначається вмістом першого операнда регістра регістрової пам’яті з номером, який відповідає вмісту програмного лічильника процесора. З табл. 1 видно, що в команді задається тільки адреса призначення результату.

Таблиця 3

Методи адресації пам’яті універсального комп’ютера

Метод адресації	Приклад команди	Зміст команди при використанні
1	2	3
Регістрова	Load #4	RM[4] <--RM[RTC_1] Для завантаження значення з одного регістра в інший
Безпосередня	Load #4,#3	RM[4] <--3 Для присвоєння констант
Базова із зсувом	Load_adi #4,#100	RM[4] <--M[RTC_1+100] Для звернення до локальних змінних
Непряма регістрова	Load_r #4	RM[4] <--M[(RTC_1)] Для звернення по покажчику або обчисленій адресі

Продовження табл. 3

1	2	3
Регістрово-індексна з поновленням регістра	Load_add #4	RM[4] <--M[RTC_1+RTC_2] Іноді корисна при роботі з масивами: RTC_1- база, RTC_2- індекс RM[4] <--RTC_1+RTC_2
Пряма або абсолютна	Load_i #4, #1000	RM[4] <--M[1000] Іноді корисна для звернення до статичних даних
PC-relative	Branch #100	PC<--PC+100(якщо виконується умова)

Висновки. Проаналізовано існуючі методи адресації в комп'ютерних архітектурах CISC та RISC. Показано, що архітектури CISC та RISC мають низку недоліків.

Запропоновано достатньо прості методи адресації, які позбавлені описаних в статті недоліків архітектур CISC та RISC, на основі яких існує можливість реалізації складніших і рідко вживаних методів адресації універсального комп'ютера.

Показано, що на основі запропонованих методів адресації за рахунок спрощеного опису команд спрощується декодування команд та зменшується складність апаратури, необхідної для декодування.

1. Шнитман В. Современные высокопроизводительные компьютеры, информационно-аналитические материалы. – Центр Информационных Технологий, 1996. 2. Орлов С., Цилькер Б. Организация ЭВМ и систем: Учеб. для вузов. – СПб.: Питер, 2007. – 672 с. 3. http://www.msclub.ce.cctpu.edu.ru. 4. Jones D.W. The Ultimate RISC // Computer Architecture News. – June 1988. – Vol. 16:3. – P. 48–55. 5. Jones D.W. A Minmal CISC // Software-Practice and Experience. – March 1988. – Vol. 18:3. – P. 56–63. 6. http://uk.wikipedia.org. 7. http://www.cs.swan.ac.uk. 8. Атовмян И.О. Архитектура вычислительных систем. – М.: Изд. МИФИ, 2002. 9. Брайдо В.Л., Ильина О.П. Архитектура ЭВМ и систем. – СПб: Питер, 2006. – 718 с. 10. Гуров В.В., Чуканов В.О. Основы теории и организации ЭВМ. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2006. – 272 с. 11. Гуров В.В. Основы организации вычислительных машин: Учеб. пособие. – М.: МИФИ, 2004. 12. Patterson, D.A. and J.L. Hennessy Computer Organization and Design: TheHardware/Software Interface, Morgan Kaufmann, San Francisco, 1997.