МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ

Харківський національний університет внутрішніх справ

Факультет № 4

Кафедра інформаційних технологій

ТЕКСТ ЛЕКЦІЇ

з дисципліни «Операційні системи та комп'ютерні мережі»

за темою «Взаємодія центрального процесора із зовнішніми пристроями»

Галузь знань: 12 "Інформаційні технології "

Спеціальність: 125 "Кібербезпека"

Ступінь вищої освіти - бакалавр

м. Харків
2017 р.

Передмова

СХВАЛЕНО
Науково-методичною радою ХНУВС
_____ Протокол № _____
(дата, місяць, рік)

ЗАТВЕРДЖЕНО
Вченою радою факультету № 4 ХНУВС
_____ Протокол № _____
(дата, місяць, рік)

_____ _____
(підпис) (П.І.Б.)

ПОГОДЖЕНО
Секцією Науково-методичної ради ХНУВС з технічних дисциплін
_____ Протокол № _____
(дата, місяць, рік)

_____ _____
(підпис) (П.І.Б.)

ЗАТВЕРДЖЕНО
На засіданні кафедри інформаційних технологій
_____ Протокол № _____
(дата, місяць, рік)

_____ _____
(підпис) (П.І.Б.)

Рецензент:
Зацеркляний М.М., доктор технічних наук, професор;

Розробники: Можаєв О.О. – Харків: Харківський національний університет внутрішніх справ, 2017

План лекції

1. Процесори з повним і скороченим набором команд.
2. Процес передачі команди від процесора до зовнішнього пристрою.
3. Реалізація програми центральним процесором.

Література:

Основна:

1. Барановская Т.П. Архитектура компьютерных систем и сетей: Учеб. пособие / Т.П. Барановская, В.И. Лойко и др.; под ред. В.И. Лойко. – М.: Финансы и статистика, 2003. – 256с.
2. Пятибратов А. П. Вычислительные системы, сети и телекоммуникации: Учебник. – 2-е изд., перераб. и доп. / А. П. Пятибратов, Л. П. Гудыно, А. А. Кириченко; Под ред. А. П. Пятибратова – М.: Финансы и статистика, 2004. – 512с.
3. Струков В.М. Комп'ютерні основи систем кібербезпеки/ Зацеркляний М.М., Струков В.М. – Харків,ХНУВС, 2017. –274с.
4. Олифер В. Г.Компьютерные сети. Принципы, технологии, протоколы / В. Г. Олифер, Н. А. Олифер. – СПб.: Питер, 2001. – 672 с.
5. Цилькер Б. Я. Организация ЭВМ и систем / Б. Я.Цилькер, С.А.Орлов. СПб.: Питер, 2006. - 668 с.
6. Столлингс В. Структурная организация и архитектура компьютерных систем. / В.Столлингс - М.: Издательский дом "Вильямс", 2002. - 896с.

Текст лекції

1. Процесори з повним і скороченим набором команд.

Невідповідність стандартів підходів, принципів мови високого рівня і обчислювальної машини Фон-Неймана та мови низького рівня називається семантичним розривом. Подолати такий розрив намагаються шляхом розширення системи команд, доповнюючи цю систему командами, які реалізують складні оператори мови високого рівня, але на апаратному рівні. Обчислювальні машини, де реалізовані такі засоби, тобто ***машини з повним набором команд*** називаються - ***CISC*** (Complex Instruction Set Computer).

Спроби за допомогою CISC подолати семантичний розрив призводять до ускладнення архітектури обчислювальної машини, особливо пристрою управління, що позначається на продуктивності машини в цілому. У CISC складно організувати ефективний конвеєр команд, який є одним із найбільш перспективних шляхів збільшення продуктивності обчислювальної машини. Аналіз програм, одержаних після компіляції з мови високого рівня, визначив такі закономірності: реалізація складних команд, еквівалентних операторам

мови високого рівня, вимагає збільшення ємності управління пам'яттю в мікропрограмному пристрої управління. Мікропрограми складних команд можуть займати до 60% обсягу ПЗП.

У скомпільованій програмі оператори мови високого рівня організовані у вигляді процедур, тому на виконання виклику процедури і повернення з неї припадає від 15% до 45% обчислювального навантаження. При виклику процедури викликаюча програма передає їй деяку кількість аргументів. У 98% випадків їх число не перевищує 6-ти. Приблизно таке ж становище складається і з параметрами, які процедура повертає. Більше 80% змінних, використовуваних програмою, локальні.

Майже половину операцій у ході обчислень становить операція присвоювання. Аналіз наведених результатів вимагає перегляду традиційних архітектурних рішень, наслідком чого є поява RISC-архітектури.

RISC (Reduced Instruction Set Computer) характерна тим, що регістрів стає багато, команди мають однакову довжину (для конвеєра), уніфіковані канали передачі.

Головні зусилля в RISC спрямовані на побудову ефективного конвеєра команд. Всі команди дістаються з оперативної пам’яті, при цьому жодна команда не повинна перебувати в стані очікування. Ідеальним вважається варіант, коли будь-який етап циклу команди виконується протягом одного такту. Цю умову можна реалізувати для етапу вибірки. Для цього необхідно, аби всі команди мали стандартну довжину.

Уніфікація часу виконання команд більш складне завдання, тому разом зі зверненнями до регістрів існує звернення до ОП. Однакова довжина команди - це не все. Скорочення самого набору команд. У в 80%-90% випадків використовується 10%-20% команд від загального списку. До найбільш часто використовуваних дій відноситься пересилання даних, а також алгебраїчні і логічні операції. Скоротимо число команд, що мають доступ до ОП. Для цього збільшимо кількість регістрів загального призначення.

Концепція RISC архітектури зводиться до таких положень:

• виконання всіх або принаймні 75% команд за 1 цикл;

• стандартна однослівна довжина всіх команд, що дорівнює природній довжині слова даних (операндів) і ширині ШД (це допускає уніфіковану потокову обробку всіх команд);

• кількість команд не більше 128-ми (реально 70);

• мала кількість форматів команд (не більше 4-ох).

Команди використовують внутрішньопроцесорні міжрегістрові пересилання. Регістрів загального призначення мінімум 32, максимум більше 500.

До *переваг RISC* відносяться:

1. Порівняно проста структура ПУ, яка в результаті займає в RISC-процесорі не більше 10% площі кристала.

2. Висока швидкодія, пов'язана з уніфікацією набору команд і орієнтацією на потокову конвеєрну обробку.

При цьому із-за скорочення числа команд на виконання ряду функцій доведеться витрачати кілька команд. Це подовжує код команди. У середньому довжина однієї і тієї ж команди у RISC-архітектурі більша за довжину аналогічної команди у CISC в середньому на 30%. Велика кількість регістрів загального призначення. Зі збільшенням числа регістрів ускладнюється процедура їх адресації. ПУ - пристрій із жорсткою логікою.

2. Процес передачі команди від процесора до зовнішнього пристрою.

Взаємодія центрального процесора із зовнішніми пристроями передбачає виконання логічної послідовності дій, пов'язаних із пошуком пристрою, визначенням його технічного стану, обміном командами та інформацією. Ця логічна послідовність дій разом із пристроями, які її реалізують, одержала назву ***інтерфейс введення-виведення***.

Для різних пристроїв можуть використовуватися різні логічні послідовності дій, тому інтерфейсів введення-виведення може в одній і тій же ЕОМ використовуватися декілька. Якщо їх вдається звести до одного, універсального, то такий інтерфейс називається стандартним. У IBM PC є три стандартних інтерфейсу для зв'язку центрального процесора із зовнішніми пристроями: ***паралельний*** (типу **Centronics**) і два ***послідовних*** (типу **RS-232** і **USB**).

Процес передачі команди до пристрою такий:

• процесор виставляє на шину адреси адресу пристрою, на шину управління - сигнал «пошук пристрою»;

• пристрій відгукується на шину управління сигналом «збіг адреси»;

• процесор спеціальним сигналом запитує байт стану і одержує його;

• процесор розміщує на шину даних команду і сигнал на шину управління «передача команди»;

• процесор очікує від пристрою підтвердження про приймання команди. І після одержання переходить до виконання чергової команди.

Якщо при зверненні центрального процесора до зовнішнього пристрою продовження виконання основної програми центральним процесором можливе тільки після завершення операції введення-виведення, то центральний процесор, завантаживши зовнішній пристрій, переходить у стан очікування і знаходиться в ньому до тих пір, поки зовнішній пристрій не повідомить йому про закінчення обміну даними. Це призводить до простою більшості пристроїв ЕОМ, оскільки в кожний момент часу може працювати тільки один із них. Такий ***режим роботи*** одержав назву ***однопрограмного*** - в кожний момент часу всі пристрої знаходяться в стані очікування і тільки один пристрій виконує основну (і єдину) програму.

Для ліквідації таких простоїв і підвищення ефективності роботи обладнання зовнішні пристрої виконані автономними. Одержавши від центрального процесора необхідну інформацію, вони самостійно організовують свою роботу з обміну даними. Процесор, завантаживши зовнішній пристрій, намагається продовжити виконання програми. В разі

необхідності (якщо зустрінуться відповідні команди) він може завантажити в роботу кілька інших пристроїв (оскільки зовнішні пристрої працюють значно повільніше процесора). Якщо ж йому доводиться переходити в режим очікування, то, користуючись тим, що в оперативній пам’яті може одночасно перебуватися не одна, а кілька програм, центральний процесор переходить до виконання чергової програми. При цьому створюється ситуація, коли в один і той же момент часу різні пристрої ЕОМ виконують або різні програми, або різні частини однієї і тієї ж програми. Такий ***режим роботи*** ЕОМ називається ***багатопрограмним***.

3.Реалізація програми центральним процесором

Програма в ЕОМ реалізується центральним процесором за допомогою послідовного виконання команд, що утворюють цю програму. Дії, необхідні для вибірки (діставання з основної пам'яті) і виконання команди, називається циклом команди. У загальному випадку цикл команди включає в себе кілька складових (етапів):

- вибірку команди;
- формування адреси наступної команди;
- декодування команди;
- обчислення адрес операндів;
- вибірку операндів;
- виконання операції;
- формування ознаки результату;
- запис результату.

Перераховані етапи виконання команди називаються ***стандартним циклом команди***. Зазначимо, що не всі етапи присутні при виконанні будь-якої команди (залежить від типу команди), проте етапи вибірки, декодування, формування адреси наступної команди і виконання операції мають місце завжди.

У певних ситуаціях можливі ще два етапи:

- непряма адресація;
- реакція на переривання.

Коротко охарактеризуємо кожний із перерахованих етапів стандартного циклу команди. Тут варто враховувати, що опис, який приводиться, має на меті лише дати уявлення про сутність кожного з етапів. У той же час розподіл функцій за різними етапами циклу команди і послідовність виконання деяких з них у реальних ЕОМ можуть відрізнятися від викладеного.

Цикл будь-якої команди розпочинається з того, що центральний процесор дістає команду з пам'яті, використовуючи адресу, що зберігається в лічильнику команд. Двійковий код команди міститься в регістрі команди і з цього моменту стає «видимим» для процесора. Якщо довжина команди збігається з розрядністю комірки пам'яті, то все зрозуміло. Проте, система команд багатьох ЕОМ передбачає кілька форматів команд, причому в різних

форматах команда може займати 1, 2 чи більше комірок, а етап вибірки команди можна вважати завершеним лише після того, як в регістрі команди розміщується повний код команди. Інформація про фактичну довжину команди міститься в полях коду операції та способу адресації. Як правило, ці поля розташовуються у першому слові коду команди і для з'ясування необхідності продовження процесу вибірки необхідне попереднє декодування їх вмісту. Таке декодування може бути виконаним після того, як перше слово коду команди опиниться в регістрі команд. У разі багатослівного формату команди процес вибірки триває аж до занесення в регістр команд усіх слів команди.

Для більшості ЕОМ характерне розміщення сусідніх команд програми в суміжних комірках пам'яті. Якщо взята команда не порушує природного порядку виконання програми, то для обчислення адреси наступної виконуваної команди достатньо збільшити вміст лічильника команд на довжину поточної команди, подану кількістю зайнятих кодом команди комірок пам'яті. Довжина команди, а також те, чи здатна вона змінити природний порядок виконання команд програми, з'ясовуються в ході попереднього декодування. Якщо взята команда здатна змінити послідовність виконання програми (команда умовного чи безумовного переходу, виклику процедури тощо), процес формування адреси наступної команди переноситься на етап виконання операції. Із-за цього у ряді ЕОМ розглянутий етап циклу команди знаходиться не за вибіркою команди, а в кінці циклу.

Після вибірки команда декодується, для чого центральний процесор розшифровує код команди, що знаходиться в регістрі команд. В результаті декодування з'ясовуються такі питання:

• чи знаходиться в регістрі команд повний код команди чи потрібне дозавантаження інших слів команди;

• які подальші дії потрібні для виконання даної команди;

• якщо команда використовує операнди, то звідки вони повинні бути взяті (номер регістра або адресу комірки основної пам'яті);

• якщо команда формує результат, то куди цей результат повинен бути спрямований.

Відповіді на два перших запитання дає розшифровка коду операції, результатом якої може бути унітарний код, де кожний розряд відповідає одній із команд. На практиці замість унітарного коду можуть зустрітися найрізноманітніші форми подання результатів декодування, наприклад адреса комірки спеціальної управляючої пам'яті, де зберігається перша мікрокоманда мікропрограми для реалізації зазначеної в команді операції.

Повне з'ясування всіх аспектів команди, крім розшифровки коду операції, вимагає також аналізу адресної частини команди, включаючи поле способу адресації.

За результатами декодування проводиться підготовка електронних схем ЕОМ до виконання запропонованих командою дій.

Етап обчислення адрес операндів має місце, якщо в процесі декодування

команди з'ясовується, що команда використовує операнди. Якщо операнди розміщуються в основній пам'яті, здійснюється обчислення їх виконавчих адрес з урахуванням зазначеного в команді способу адресації. Так, у разі індексної адресації для одержання виконавчої адреси проводиться підсумовування вмісту адресної частини команди і вмісту індексного регістра.

Обчислені виконавчі адреси використовуються для зчитування операндів із пам'яті і занесення їх у певні регістри процесора. Наприклад, у разі арифметичної команди операнд після діставання з пам'яті може бути завантаженим у вхідний регістр арифметико-логічного пристрою. Проте частіше операнди попередньо заносяться в спеціальні допоміжні регістри процесора, а їх пересилання на вхід арифметико-логічного пристрою відбувається на етапі виконання операції.

На етапі виконання операції реалізується зазначена в команді операція. Із-за відмінності сутності кожної команди ЕОМ зміст цього етапу суто індивідуальний.

На етапі формування ознаки результату визначається, яким дістався результат операції. Результат може бути додатним, від'ємним, рівним нулю і т.п. Сформована ознака заноситься в регістр ознаки результату для подальшого використання пристроєм управління.

Етап запису результату присутній у циклі тих команд, які допускають занесення результату в регістр або комірку основної пам'яті. Фактично його можна вважати частиною етапу виконання, особливо для тих команд, які розміщують результат відразу в кілька місць.

Чимало команд допускають читання операндів із пам'яті або запис у пам'ять. У найпростішому випадку в адресному полі таких команд явно вказується виконавча адреса відповідної комірки ОП. Проте часто використовується й інший спосіб вказівки адреси, коли адреса операнда зберігається в якійсь комірці пам'яті, а в команді вказується адреса комірки, що містить адресу операнда. Подібний прийом називається ***непрямою адресацією***. Аби прочитати або записати операнд, спочатку потрібно дістати з пам'яті його адресу і тільки після цього провести потрібну дію (читання або запис операнда), іншими словами, потрібно виконати два звернення до пам'яті. Це, природно, відображається і на циклі команди, в якому з'являється непряма адресація. Етап непрямої адресації можна віднести до етапу обчислення адрес операндів, оскільки його сутність зводиться до визначення виконавчої адреси операнда. Іншими словами, вміст адресного поля команди в регістрі команд використовується для звернення до комірки ОП, в якій зберігається адреса операнда, після чого одержана з пам'яті виконавча адреса операнда розміщується в адресне поле регістра команди на місце непрямої адреси. Подальше виконання команди протікає стандартним чином.

Спосіб адресації – це найважливіша характеристика архітектури обчислювальної машини, яка дозволяє або не дозволяє вирішити такі суперечності: з одного боку (з точки зору скорочення апаратних витрат) прагнення зменшити довжину адресного поля, з іншого боку спосіб задання

адреси повинен сприяти максимальному зближенню конструктивних характеристик мов програмування високого рівня і машинних команд. Вирішення цієї суперечності і призвело до використання різних способів адресації.

При ***безпосередній адресації*** в адресному полі міститься сам операнд. Застосовується такий спосіб при виконанні арифметичних операцій, операцій порівняння, завантаження констант у регістри. Перевагою такого способу адресації є швидкість виконання, оскільки немає звернення в оперативну пам'ять. Недоліки такого способу: розмір операнда обмежений довжиною адресного поля команди.

При ***прямій адресації*** адресний код ($A_к$) прямо вказує номер комірки пам'яті, до якої проводиться звернення. Виконавчий код ($A_{вик}$) збігається з адресним кодом. Недоліком такого способу адресації є недостатні можливості при зверненні до адресного простору великого розміру. Адреса вказана в команді не може бути зміненою в процесі обчислень.

Непряма адресація – це адресація адреси. Її переваги: дозволяє подолати труднощі, які є у прямій і безпосередній адресації. Недолік полягає у подвійному зверненні до пам'яті (погіршує часові характеристики, витрачається 2 комірки оперативної пам'яті).

Регістрова адресація нагадує пряму адресацію. Різниця в тому, що адресне поле вказує не на комірку пам'яті, а на регістр процесора. Як правило, розмір адресного поля дорівнює 3 і 4 бітам, що дозволяє адресувати 8 і 16 регістрів загального призначення. Переваги реєстрової адресації такі:

1) коротке адресне полі команди;

2) виключаються звернення до оперативної пам'яті.

Недоліки: Обмежене число регістрів загального призначення не дозволяє широко використовувати цю адресацію.

При ***непрямій регістровій адресації*** виконавча адреса операнда зберігається не в комірці основної пам'яті, а в регістрі процесора. Переваги: такі ж можливості, як і у непрямій адресації, але на одне звернення до пам'яті менше. Недоліки: обмежене число регістрів загального призначення не дозволяє широко використовувати цю адресацію.

При ***відносній адресації*** виконавча адреса визначається сумою адресного коду і деякого числа, яке називається базовою адресою. Для зберігання базових адрес в обчислювальній машині можуть передбачатися спеціальні регістри або виділені комірки пам'яті.

У команді в адресному полі виділяється підполе для зазначення номера базового регістра. У ньому знаходиться адреса першої комірки масиву. Залишена частина адресного поля містить адресний код, який використовується для подання порівняно короткого зміщення відносно початку масиву.

Найчастіше виконавча адреса при базуванні утворюється за допомогою суматора. Відносна адресація використовується для доступу до елементів масиву, знаходження якого в пам'яті в процесі обчислень може змінюватися.

Зміщення має меншу довжину, ніж повна адреса, і це дозволяє скоротити довжину адресного поля команди, а отже, і саму команду.

Індексна адресація . Характерним для реалізації в машині розв’язування математичних задач є циклічність обчислювальних процесів. Це означає, що одна і та ж команда виконується, але над різними операндами, розташованими впорядковано в пам'яті.

Програмування обчислювальних циклів спрощується, якщо після кожного циклу забезпечується автоматична зміна відповідно команд і їх адресних частин. Причому зміна така, що узгоджується з розташуванням у пам'яті виконавчих операндів. Індексна адресація є зручним механізмом для організації циклічних обчислень.