ОСОБЛИВОСТІ ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ 32-РОЗРЯДНОГО АСЕМБЛЕРА

Укладачі: Мархивка В.С., ст. викладач,
Олексів М. В., асистент, к.т.н.,
Мороз І.В., ст. викладач, к.т.н.,
Акимишин О.І., доцент, к.т.н..

ТЕОРЕТИЧНІ ВІДОМОСТІ

Основою для розробки низькорівневого системного програмного забезпечення є програмна модель комп'ютера, частиною якої є ***програмна модель мікропроцесора***. До складу програмної моделі мікропроцесорів Intel сімейства x86 входять 32 регістри в тій чи іншій мірі доступні для використання програмістом. Дані регістри можна розділити на дві великі групи:

- 16 регістрів користувача;
- 16 системних регістрів.

У програмах на мові асемблера регістри використовуються дуже інтенсивно. Більшість регістрів мають певне функціональне призначення.

Регістри користувача

Як випливає з назви, *призначеними для користувача* регістри називаються тому, що програміст може використовувати їх при написанні своїх програм. До цих регістрів відносяться (рис.1.1):

- вісім 32-бітових регістрів, які можуть використовуватися програмістами для зберігання даних і адрес (їх ще називають регістрами загального призначення (РЗП)):
  - **eax/ax/ah/al**;
  - **ebx/bx/bh/bl**;
  - **edx/dx/dh/dl**;
  - **ecx/cx/ch/cl**;
  - **ebp/bp**;
  - **esi/si**;
  - **edi/di**;
  - **esp/sp**.
- шість сегментних регістрів: **cs**, **ds**, **ss**, **es**, **fs**, **gs**;
- регістри управління та стану:
  - регістр прапорів **eflags**/**flags**;
  - регістр покажчика команди **eip**/**ip**.

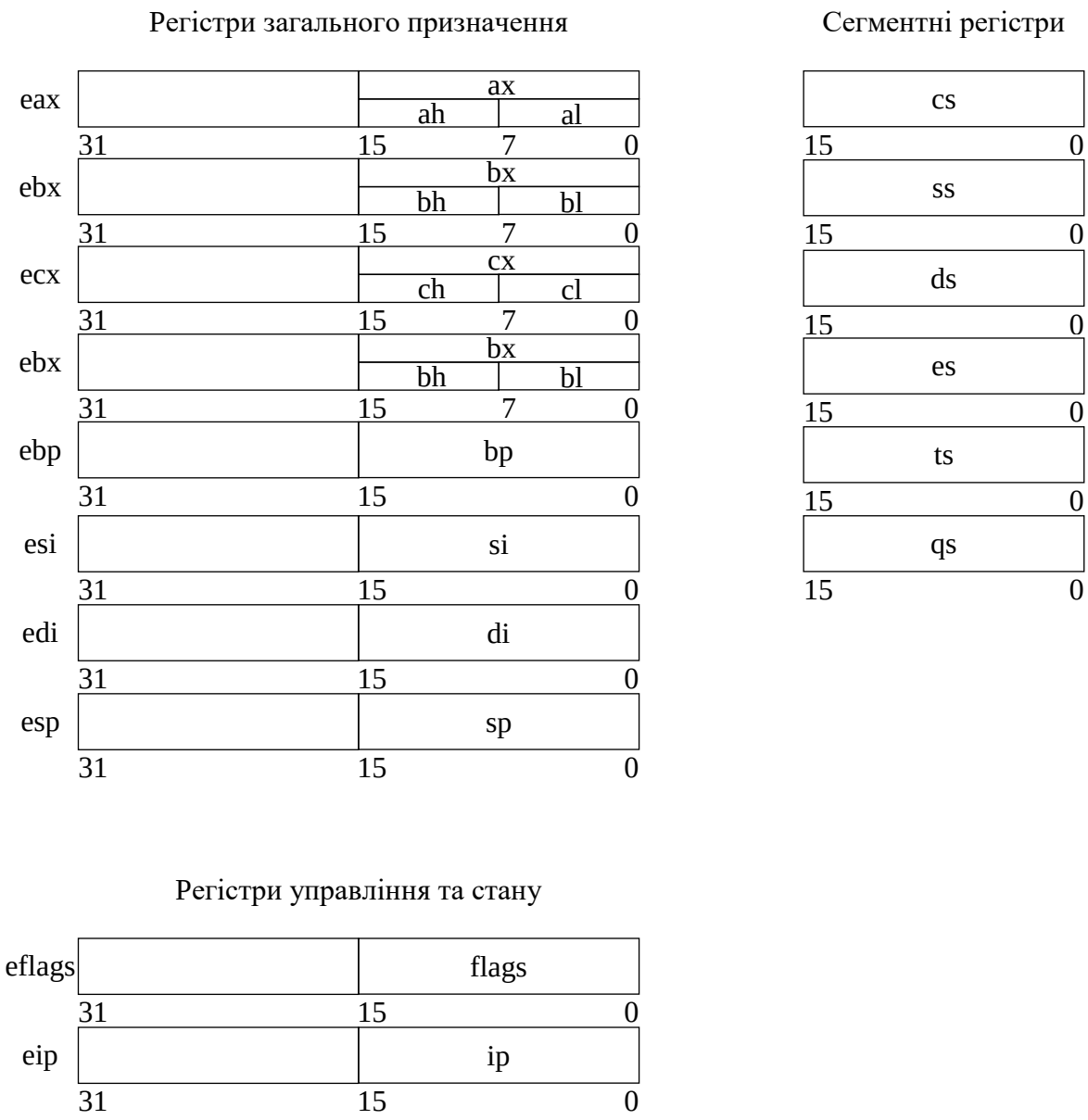

Рис. 1.1. Регістри користувача мікропроцесорів i486 і Pentium

Як видно з рис. 1.1, частина регістрів зображені розділено. Це не різні регістри — це частини одного великого 32-розрядного регістру. Їх можна використовувати в програмі як окремі об’єкти. Так зроблено, для забезпечення сумісності з програмами, написаними для ранніх, 16-розрядних, моделей мікропроцесорів фірми Intel, починаючи з i8086. Мікропроцесори i486 і Pentium мають в основному 32-розрядні регістри. Їх кількість, за винятком сегментних регістрів, така ж, як і у i8086, але розмірність більше, що і відображено в їх позначеннях — вони мають приставку **e** (***Extended***).

Регістри загального призначення

Всі регістри цієї групи дозволяють звертатися до своїх “молодших” частин (див. рис. 1.1). Слід відзначити, що використовувати для самостійної адресації можна тільки молодші 16- і 8-бітові частини цих регістрів. Старші 16 біт цих регістрів як самостійні об’єкти програмування недоступні. До РЗП відносяться:

- **eax/ax/ah/al** (Accumulator register) — акумулятор. Використовується для зберігання проміжних даних. У деяких командах використання цього регістра обов’язкове;
- **ebx/bx/bh/bl** (Base register) — *базовий* регістр. Зазвичай застосовується для зберігання базової адреси деякого об’єкту в пам’яті;
- **ecx/cx/ch/cl** (Count register) — *регістр-лічильник*. Застосовується в командах, що проводять деякі дії, що повторюються. Його використання часто неявно і приховано в алгоритмі роботи відповідної команди. Наприклад, команда організації циклу `loop` окрім передачі управління команді, що знаходиться за деякою адресою, аналізує і зменшує на одиницю значення регістра ecx/cx;
- **edx/dx/dh/dl** (Data register) — регістр *даних*. Так само, як і регістр eax/ax/ah/al, зберігає проміжні дані. У деяких командах його використання обов’язкове, а для деяких це відбувається неявно.

Наступні два регістри використовуються для підтримки операцій, що проводять послідовну обробку ланцюжків елементів, кожний з яких може мати довжину 32, 16 або 8 біт:

- **esi/si** (Source Index register) — *індекс джерела*.
- **edi/di** (Destination Index register) — *індекс приймача* (одержувача).

В архітектурі мікропроцесора на програмно-апаратному рівні підтримується така структура даних, як ***стек***. Для роботи із стеком в системі команд мікропроцесора є спеціальні команди, а в програмній моделі мікропроцесора для цього існують спеціальні регістри:

- **esp/sp** (Stack Pointer register) — регістр *покажчика стеку*. Містить покажчик вершини стеку в поточному сегменті стеку.
- **ebp/bp** (Base Pointer register) — регістр *покажчика бази кадру стеку*. Призначений для організації довільного доступу до даних усередині стеку.

Жорстке закріплення регістрів для деяких команд дозволяє компактніше кодувати їх машинне подання. Знання цих особливостей дозволяє при необхідності хоч би на декілька байтів заощадити пам’ять, що займається кодом програми.

Робота з масивами

Регістри загального призначення використовуються для адресації масивів. При цьому застосовується адресація за базою з масштабуванням: початкова адреса + база * масштабуючий коефіцієнт бази. Допустимі значення масштабуючого коефіцієнту бази (кількість байтів, які займає 1 елемент масиву) рівні 1, 2, 4, 8.

Таким чином, щоб записати 3-й елемент масиву оголошеного мовою С як `short arr[15]` в `edx` треба написати код:

```
mov eax, arr
mov ebx, 2
mov edx,[eax+ebx*2]
```

Сегментні регістри

У програмній моделі мікропроцесора є шість сегментних регістрів: *cs*, *ss*, *ds*, *es*, *gs* та *fs*. Їх існування обумовлено специфікою організації і використання оперативної пам'яті мікропроцесорами Intel. Вона полягає в тому, що мікропроцесор апаратно підтримує структурну організацію програми у вигляді трьох частин, що називаються сегментами. Відповідно, така організація пам'яті називається ***сегментною***.

Для того, щоб вказати на сегменти, до яких програма має доступ в конкретний момент часу, і призначені *сегментні регістри*. Фактично, з невеликою поправкою, як показано далі, в цих регістрах містяться адреси пам'яті з яких починаються відповідні сегменти. Логіка обробки машинної команди побудована так, що при вибірці команди доступу до даних програми або до стеку неявно використовуються адреси з певних сегментних регістрів. Мікропроцесор підтримує наступні типи сегментів:

1. ***Сегмент коду***. Містить команди програми. Для доступу до цього сегменту служить регістр **cs** (code segment register) — *сегментний регістр коду*. Він містить адресу сегменту з машинними командами, до якого має доступ мікропроцесор (тобто ці команди завантажуються в конвейєр мікропроцесора).
2. ***Сегмент даних***. Містить оброблювані програмою дані. Для доступу до цього сегменту служить регістр **ds** (data segment register) — *сегментний регістр даних*, який зберігає адресу сегменту даних поточної програми.
3. ***Сегмент стеку***. Цей сегмент є ділянкою пам'яті, що називається *стеком*. Роботу із стеком мікропроцесор організовує за наступним принципом: останній *записаний в цю ділянку елемент вибирається першим.* Для доступу до цього сегменту служить регістр ss (stack segment register) — сегментний *регістр стеку,* що містить адресу сегменту стеку.

4. ***Додатковий сегмент даних***. Неявно алгоритми виконання більшості машинних команд припускають, що оброблювані ними дані розташовані в сегменті даних, адреса якого знаходиться в сегментному регістрі *ds*. Якщо програмі недостатньо одного сегменту даних, то вона має можливість використовувати ще три додаткові сегменти даних. Але на відміну від основного сегменту даних, адреса якого міститься в сегментному регістрі ds, при використанні додаткових сегментів даних їх адреси потрібно указувати явно за допомогою спеціальних префіксів перевизначення сегментів в команді. Адреси додаткових сегментів даних повинні міститися в регістрах **es, gs, fs** (extension data segment registers).

Регістри стану і управління

У мікропроцесор включені декілька регістрів (див. рис. 1.1), які постійно містять інформацію про стан як самого мікропроцесора, так і програми, команди якої в даний момент завантажені в конвеєр. До цих регістрів відносяться:

- регістр прапорів **eflags/flags**;
- регістр покажчика команди **eip/ip**.

Використовуючи ці регістри, можна одержувати інформацію про результати виконання команд і впливати на стан самого мікропроцесора. Розглянемо докладніше призначення і вміст цих регістрів.

eflags/flags (flag register) — регістр *прапорів*. Розрядність eflags/flags — 32/16 бітів. Окремі біти даного регістра мають певне функціональне призначення і називаються прапорцями. Молодша частина цього регістра повністю аналогічна регістру flags для i8086. На рис. 1.2 показано вміст регістра *eflags*.

Виходячи з особливостей використання, прапори регістра *eflags/flags* можна розділити на три групи:

- *8 прапорців стану*. Ці прапорці можуть змінюватися після виконання машинних команд. Прапорці стану регістра *eflags* відображають особливості результату виконання арифметичних або логічних операцій. Це дає можливість аналізувати стан обчислювального процесу і реагувати на нього за допомогою команд умовних переходів і викликів підпрограм.
- *1 прапорець управління*. Позначається **df** (Directory Flag). Він знаходиться в 10-му біті регістру *eflags* і використовується для операцій рядками даних. Значення прапорця *df* визначає напрям поелементної обробки в цих операціях: від початку рядка до кінця або навпаки, від кінця рядка до його початку (df = 1). Для роботи з прапорцем *df* існують спеціальні команди: `cld` (зняти прапорець *df*) і `std` (встановити прапорець *df*). Застосування

цих команд дозволяє привести прапорець *df* у відповідність з алгоритмом і забезпечити автоматичне збільшення або зменшення лічильників при виконанні операцій з рядками;

- *5 системних прапорців*, для керування вводом/виводом, маскованими перериваннями, відлагодженням, перемиканням між завданнями і віртуальним режимом 8086. Прикладним програмам не рекомендується модифікувати без необхідності ці прапорці, оскільки в більшості випадків це приведе до переривання роботи програми.

Рис. 1.2. Вміст регістра eflags

eip/ip (Instraction Pointer register) — регістр-покажчик *команд*. Регістр *eip*/*ip* має розрядність 32/16 бітів і містить зміщення до наступної команди, що має виконуватись, відносно адреси записаної в сегментний регістр *cs*. Цей регістр безпосередньо недоступний програмісту, але завантаження і зміна його значення проводяться різними командами управління, до яких відносяться команди умовних і безумовних переходів, виклику і повернення з процедур. Виникнення переривань також приводить до модифікації регістра *eip*/*ip*.

Системні регістри мікропроцесора

Сама назва цих регістрів говорить про те, що вони виконують специфічні функції в системі. Використання системних регістрів жорстко регламентовано. Саме вони забезпечують роботу захищеного режиму. Їх також можна розглядати як частина архітектури мікропроцесора, яка навмисно залишена

видимою для того, щоб кваліфікований системний програміст міг виконати самі низькорівневі операції. Системні регістри можна розділити на три групи:

- чотири регістри управління;
- чотири регістри системних адрес;
- вісім регістрів відлагодження.

Регістри управління

До групи регістрів управління входять 4 регістри: **cr0, cr1, cr2, cr3**.

Ці регістри призначені для загального управління системою. Регістри управління доступні тільки програмам з рівнем привілеїв 0.

Хоча мікропроцесор має чотири регістри управління, доступними є тільки три з них — виключаючи ***cr1***, функції якого поки не визначені (він зарезервований для майбутнього використання).

Регістр **cr0** містить *системні прапорці*, для керування режимами роботи мікропроцесора, що відображають його стан глобально, незалежно від конкретних завдань, що виконуються.

Призначення системних прапорців:

- ***pe*** (Protect Enable), біт 0 — *дозвіл захищеного режиму роботи*. Стан цього прапорця показує, в якому з двох режимів — реальному (pe = 0) або захищеному (pe = 1) — працює мікропроцесор в даний момент часу.
- ***mp*** (Math Present), біт 1 — *наявність співпроцесора*. Завжди 1.
- ***ts*** (Task Switched), біт 3 — *перемикання завдань*. Процесор автоматично встановлює цей біт при перемиканні на виконання іншого завдання.
- ***am*** (Aligment Mask), біт 18 — *маска вирівнювання*. Цей біт дозволяє (am = 1) або забороняє (am = 0) контроль вирівнювання.
- ***cd*** (Cache Disable), біт 30, — *заборона кеш-пам'яті*. За допомогою цього біта можна заборонити (cd = 1) або дозволити (cd = 0) використання внутрішньої кеш-пам'яті (кеш-пам'яті першого рівня).
- ***pg*** (PaGing), біт 31, — дозвіл (pg = 1) або заборона (pg = 0) *сторінкового перетворення*.

Регістр **cr2** використовується при сторінковій організації оперативної пам'яті для реєстрації ситуації, коли поточна команда звернулася за адресою, що міститься в сторінці пам'яті, відсутній в даний момент часу в пам'яті.

У такій ситуації в мікропроцесорі виникає виняткова ситуація з номером 14, і лінійна 32-бітова адреса команди, що викликала цей виняток, записується в регістр *cr2*. Маючи цю інформацію, обробник винятоку 14 визначає потрібну сторінку, здійснює її підкачку в пам'ять і відновлює нормальну роботу програми.

Регістр **cr3** також використовується при сторінковій організації пам’яті. Це так званий *регістр каталогу сторінок першого рівня*. Він містить 20-бітову фізичну базову адресу каталогу сторінок поточного завдання. Цей каталог містить 1024 32-бітових дескриптора, кожний з яких містить адресу таблиці сторінок другого рівня. У свою чергу кожна з таблиць сторінок другого рівня містить 1024 32-бітових дескриптора, що адресують сторінкові кадри в пам’яті. Розмір сторінкового кадру — 4 Кбайт.

Регістри системних адрес

Ці регістри ще називають *регістрами управління пам’яттю*. Вони призначені для захисту програм і даних в мультизадачному режимі роботи мікропроцесора. При роботі в захищеному режимі мікропроцесора адресний простір ділиться на:

- *глобальний* — загальний для всіх завдань;
- *локальний* — окремий для кожного завдання.

Цим розділенням і пояснюється присутність в архітектурі мікропроцесора наступних системних регістрів:

- *регістра таблиці глобальних дескрипторів* **gdtr** (Global Descriptor Table Register) що має розмір 48 біт і що містить 32-бітову (биті 16—47) базову адресу глобальної дескрипторної таблиці GDT і 16-бітове (биті 0—15) значення межі, що є розміром в байтах таблиці GDT;
- *регістра таблиці локальних дескрипторів* **ldtr** (Local Descriptor Table Register) що має розмір 16 біт і що містить так званий селектор дескриптора локальної дескрипторної таблиці LDT. Цей селектор є покажчиком в таблиці GDT, який і описує сегмент, що містить локальну дескрипторну таблицю LDT;
- *регістра таблиці дескрипторів переривань* **idtr** (Interrupt Descriptor Table Register) що має розмір 48 біт і що містить 32-бітову (биті 16–47) базову адресу дескрипторної таблиці переривань IDT і 16-бітове (биті 0—15) значення межі, що є розміром в байтах таблиці IDT;
- *16-бітового регістра завдання* **tr** (Task Register), який подібно до регістру ldtr, містить селектор, тобто покажчик на дескриптор в таблиці GDT. Цей дескриптор описує *поточний сегмент стану завдання* (TSS — Task Segment Status). Цей сегмент створюється для кожного завдання в системі, має жорстко регламентовану структуру і містить контекст (поточний стан) завдання. Основне призначення сегментів TSS — зберігати поточний стан завдання у момент перемикання на інше завдання.

Регістри відлагодження

Це дуже цікава група регістрів, призначених для апаратного відлагодження. Засоби апаратного відлагодження вперше з'явилися в мікропроцесорі i486. Мікропроцесор містить вісім регістрів відлагодження, але реально з них використовуються тільки 6.

Регістри **dr0, dr1, dr2, dr3** мають розрядність 32 біти і призначені для завдання лінійних адрес чотирьох точок переривання. Регістр **dr6** називається регістром стану відлагодження. Біти цього регістра встановлюються відповідно до причин, які викликали виникнення останнього винятку з номером 1. Перерахуємо ці біти і їх призначення:

- ***b0*** — якщо цей біт встановлений в 1, то останній виняток (переривання) виникло в результаті досягнення контрольної крапки, визначеної в регістрі *dr0*;
- ***b1*** — аналогічно *b0*, але для контрольної крапки в регістрі dr1;
- ***b2*** — аналогічно *b0*, але для контрольної крапки в регістрі dr2;
- ***b3*** — аналогічно *b0*, але для контрольної крапки в регістрі dr3;
- ***bd*** (біт 13) — служить для захисту регістрів відлагодження;
- ***bs*** (біт 14) — встановлюється в 1, якщо виключення 1 було викликано станом прапорця *tf* = 1 в регістрі eflags;
- ***bt*** (біт 15) встановлюється в 1, якщо виключення 1 було викликано перемиканням на завдання зі встановленим бітом пастки в TSS *t* = 1.

Решта всіх бітів в цьому регістрі заповнюється нулями. Обробник винятку 1 за вмістом *dr6* повинен визначити причину, після якої відбувся виняток, і виконати необхідні дії.

Регістр *dr7* називається регістром управління відлагодженням. У ньому для кожного з чотирьох регістрів контрольних точок відлагодження є поля, за допомогою яких можна уточнити наступні умови, при яких слід згенерувати переривання:

- *місце реєстрації контрольної крапки* — тільки в поточному завданні або в будь-якому завданні. Ці біти займають молодші вісім біт регістра dr7 (по два бита на кожну контрольну крапку (фактично точку переривання), що задається регістрами dr0, dr1, dr2, dr3 відповідно). Перший біт з кожної пари — це так званий локальний дозвіл; його установка говорить про те, що точка переривання діє якщо вона знаходиться в межах адресного простору поточного завдання. Другий біт в кожній парі визначає глобальний дозвіл, який говорить про те, що дана контрольна крапка діє в межах адресних просторів всіх завдань, що знаходяться в системі;

- *тип доступу*, за яким ініціюється переривання: тільки при вибірці команди, при записі або при записі/читанні даних. Біти, що визначають подібну природу виникнення переривання, локалізуються в старшій частині даного регістра.

КОНТРОЛЬНІ ПИТАННЯ

1. Які програмно доступні регістри архітектури IA-32 ви знаєте?
2. У чому полягає різниця між програмними моделями архітектур x86 і IA-32?
3. Як написати програму використовуючи MASM 32?
4. Як відлагодити програму використовуючи MASM 32?
5. Принципи роботи з масивами даних в Асемблері.