Лекція-1. Сегментні регістри частина 2

- У програмній моделі мікропроцесора мається шість сегментних регістрів: *cs, ss, ds, es, gs, fs*. Їхнє існування обумовлене специфікою організації і використання оперативної пам'яті мікропроцесорами. Вона полягає в тім, що мікропроцесор апаратно підтримує структурну організацію програми у виді трьох частин, називаних *сегментами*. Відповідно, така організація пам'яті називається ***сегментною***.

Сегментні регістри

- Для того щоб указати на сегменти, до яких програма має доступ у конкретний момент часу, і призначені *сегментні регістри*. Фактично, з невеликим виправленням, як ми побачимо далі, у цих регістрах містяться адреси пам'яті з який починаються відповідні сегменти. Логіка обробки машинної команди побудована так, що при вибірці команди, доступі до даних чи програми до стеку неявно використовуються адреси в цілком визначених сегментних регістрах.

Сегментні регістри

- Мікропроцесор підтримує наступні типи сегментів:
- ***Сегмент коду***. Містить команди програми. Для доступу до цього сегмента служить регістр **cs** (code segment register) — *сегментний регістр коду*. Він містить адресу сегмента з машинними командами, до якого має доступ мікропроцесор (тобто ці команди завантажуються в конвеєр мікропроцесора).

Сегментні регістри

- Мікропроцесор підтримує наступні типи сегментів:
- ***Сегмент коду***.
- ***Сегмент даних***. Містить оброблювані програмою дані. Для доступу до цього сегмента служить регістр **ds** (data segment register) — *сегментний регістр даних*, що зберігає адресу сегмента даних поточної програми.

Сегментні регістри

- Мікропроцесор підтримує наступні типи сегментів:
- ***Сегмент коду***.
- ***Сегмент даних***.
- ***Сегмент стека***. Цей сегмент являє собою область пам'яті, називану *стеком*. Роботу зі стеком мікропроцесор організує по наступному принципу: *останній записаний у цю область елемент вибирається першим*. Для доступу до цього сегмента служить регістр **ss** (stack segment register) — *сегментний регістр стека*, що містить адреса сегмента стека.

Сегментні регістри

- ***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 сегмента даних, то вона має можливість використовувати ще три додаткових сегменти даних. Але на відміну від основного сегмента даних, адреса якого міститься в сегментному регістрі *ds*, при використанні додаткових сегментів даних їхньої адреси потрібно вказувати явно за допомогою спеціальних *префіксів пере визначення сегментів* у команді. Адреси додаткових сегментів даних повинні міститися в регістрах **es, gs, fs** (extension data segment registers).

Сегментні регістри

- Мікропроцесор підтримує наступні типи сегментів:
- ***Сегмент коду***. **cs** (code segment register)
- ***Сегмент даних***. **ds** (data segment register)
- ***Сегмент стека***. **ss** (stack segment register)
- ***Додатковий сегмент даних***. **es, gs, fs** (extension data segment registers).

Регістри стану і керування

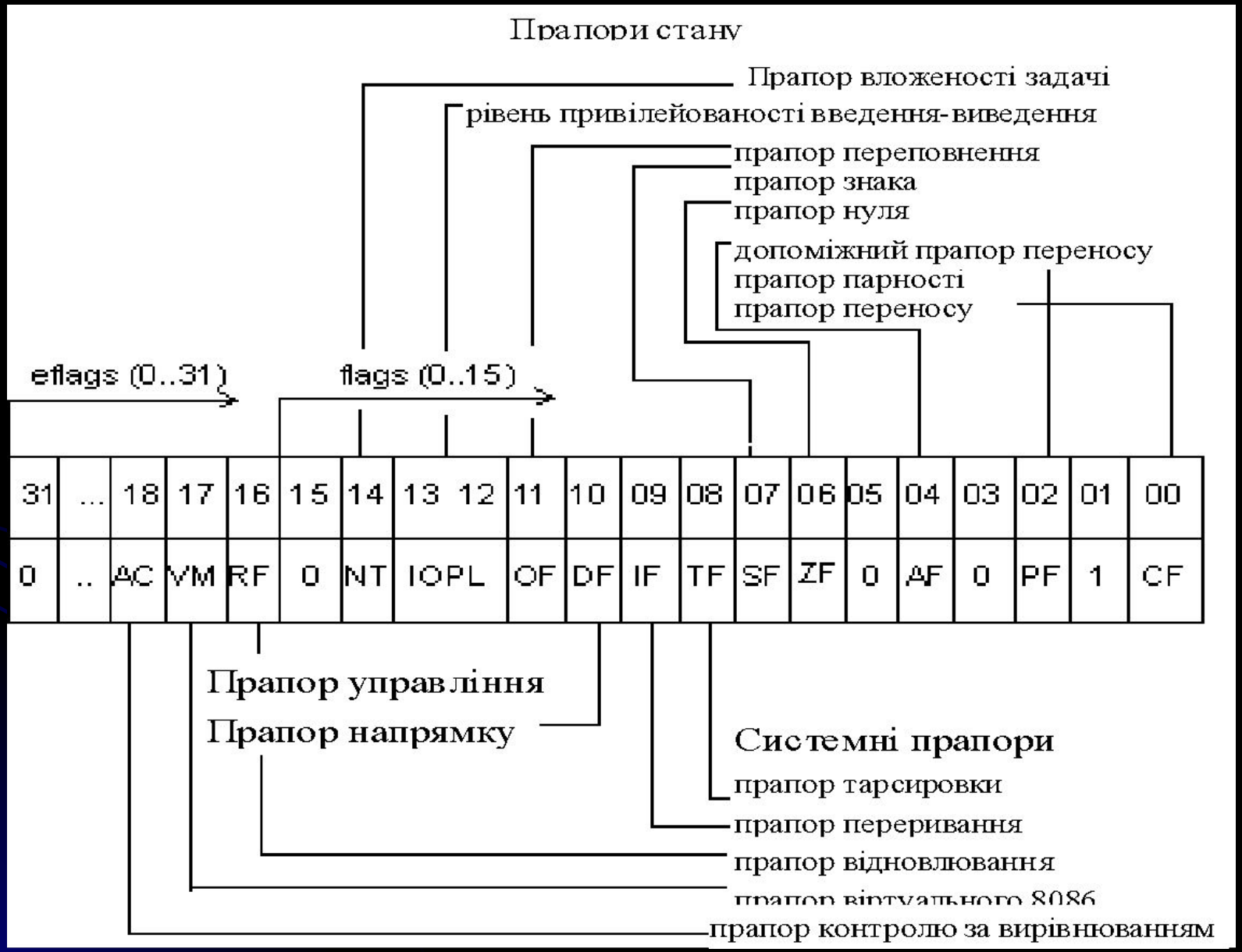
- У мікропроцесор включені кілька регістрів, які постійно містять інформацію про стан як самого мікропроцесора, так і програми, команди якого в даний момент завантажені на конвеєр. До цих регістрів відносяться:
- регістр прапорів **eflags/flags**;
- регістр покажчика команди **eip/ip**.

Використовуючи ці регістри, можна одержувати інформацію про результати виконання команд і впливати на стан самого мікропроцесора.

Регістри стану і керування

- ***eflags/flags*** (flag register) — регістр *прапорів*. Розрядність *eflags/flags* — 32/16 біт. Окремі біти даного регістра мають визначене функціональне призначення і називаються прапорами.

Регістри стану і керування



Регістри стану і керування

- ***Виходячи з особливостей використання, прапори регістра eflags/flags можна розділити на три групи:***
- *8 прапорів стану*. Ці прапори можуть змінюватися після виконання машинних команд. *Прапори стану* регістра *eflags* відбивають особливості результату виконання арифметичних чи логічних операцій. Це дає можливість аналізувати стан обчислювального процесу і реагувати на нього за допомогою команд умовних переходів і викликів підпрограм;

Регістри стану і керування

- *1 прапор керування*. Позначається **df** (Directory Flag). Він знаходиться в 10-м біті регістра *eflags* і використовується ланцюговими командами. Значення прапора *df* визначає напрямок заелементної обробки в цих операціях: від початку рядка до кінця (*df* = 0) або навпаки, від кінця рядка до його початку (*df* = 1). Для роботи з прапором *df* існують спеціальні команди: cld (зняти прапор *df*) і std (встановити прапор *df*). Застосування цих команд дозволяє привести прапор *df* у відповідність з алгоритмом і забезпечити автоматичне збільшення чи зменшення лічильників при виконанні операцій з рядками;

Регістри стану і керування

- *5 системних прапорів*, керуючих вводом/вводом, маскируемыми перериваннями, налагодженням, перемиканням між задачами і віртуальним режимом. Прикладним програмам не рекомендується модифікувати без необхідності ці прапори, тому що в більшості випадків це приведе до переривання роботи програми.

Прапори стану

Мнемоніка прапора	Прапор	Номер біта в *eflags*	Зміст і призначення
cf	Прапор переносу (Carry Flag)	0	1 — арифметична операція зробила перенос зі старшого біта результату. Старшим є 7, 15 чи 31-й біт у залежності від розмірності операнда; 0 — переносу не було

Прапори стану

Мнемоніка прапора	Прапор	Номер біта в *eflags*	Зміст і призначення
pf	Прапор паритету (Parity Flag)	2	1 — 8 молодших розрядів (цей прапор — тільки для 8 молодших розрядів операнда будь-якого розміру) результату містять парне число одиниць; 0 — 8 молодших розрядів результату містять непарне число одиниць

Прапори стану

af	Допоміжний прапор переносу (Auxiliary carry Flag)	4	Тільки для команд працюючих з BCD-числами. Фіксує факт запозичення з молодшої тетради результату: 1 — у результаті операції додавання був зроблений перенос з розряду 3 у старший розряд чи при вирахуванні була позика в розряд 3 молодшої тетради зі значення в старшої тетради; 0 — переносів і запозичень у(з) 3 розряд(а) молодшої тетради результату не було

Прапори стану

zf	Прапор нуля (Zero Flag)	6	1 — результат нульової; 0 — результат ненульовий
sf	Прапор знака (Sign Flag)	7	Відбиває стан старшого біта результату (біти 7, 15 чи 31 для 8, 16 чи 32-розрядних операндів відповідно): 1 — старший біт результату дорівнює 1; 0 — старший біт результату дорівнює 0

Прапори стану

of	Прапор переповнення (Overflow Flag)	11	Прапор of використовується для фіксування факту втрати значущого біта при арифметичних операціях: 1 — у результаті операції відбувається перенос (позика) у(з) старшого, знакового біта результату (біти 7, 15 чи 31 для 8, 16 чи 32-розрядних операндів відповідно); 0 — у результаті операції не відбувається переносу (позики) у (з) старшого, знакового біта результату

Прапори стану

iopl	Рівень Привілеїв уведення-висновку (Input/Output Privilege Level)	12, 13	Використовується в захищеному режимі роботи мікропроцесора для контролю доступу до команд введення-висновку в залежності від привілейованості задачі
nt	прапор вкладеності задачі (Nested Task)	14	Використовується в захищеному режимі роботи мікропроцесора для фіксації того факту, що одна задача вкладена в іншу

Домашнє завдання

- Вивчити зміст і призначення системних прапорів

Регістри стану і керування

- ***eip/ip*** (Instraction Pointer register) — регістр-покажчик *команд*. Регістр *eip/ip* має розрядність 32/16 біт і містить зсув наступної підлягаючої виконанню команди щодо вмісту сегментного регістра cs у поточному сегменті команд. Цей регістр безпосередньо недоступний програмісту, але завантаження і зміна його значення робляться різними командами керування, до яких відносяться команди умовних і безумовних переходів, виклику процедур і повернення з процедур. Виникнення перериванъ 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 частину архітектури мікропроцесора, що навмисно залишена видимої для того, щоб кваліфікований системний програміст міг виконати операції на найнижчім рівні.
- Системні регістри можна розділити на три групи:
- чотири регістра керування;
- чотири регістра системних адрес;
- вісім регістрів налагодження.

Регістри керування

- У групу регістрів керування входять 4 регістри: **cr0, cr1, cr2, cr3**.
- Ці регістри призначені для загального керування системою. Регістри керування доступні тільки програмам з рівнем привілеїв 0.
- Хоча мікропроцесор має чотири регістри керування, доступними є тільки три з них — виключається *cr1*, функції якого поки не визначені (він зарезервований для майбутнього використання).
- Регістр cr0 містить *системні прапори*, що керують режимами роботи мікропроцесора і відбивають його стан глобально, незалежно від конкретних задач, що виконуються. Призначення системних прапорів:
- ***pe*** (Protect Enable), біт 0 — *дозвіл захищеного режиму роботи*. Стан цього прапора показує, у якому з двох режимів — *реальному* (pe=0) чи *захищеному* (pe=1) — працює мікропроцесор у даний момент часу.
- ***mp*** (Math Present), біт 1 — *наявність співпроцесора*. Завжди 1.

Регістри керування

- ***ts*** (Task Switched), біт 3 — *переключення задач*. Процесор автоматично встановлює цей біт при переключенні на виконання іншої задачі.
- ***am*** (Aligment Mask), біт 18 — *маска вирівнювання*. Цей біт дозволяє (am = 1) чи забороняє (am = 0) контроль вирівнювання.
- ***cd*** (Cache Disable), біт 30, — *заборона кеш-пам'яті*. За допомогою цього біта можна заборонити (cd = 1) чи дозволити (cd = 0) використання внутрішньої кеш-пам'яті (кеш-пам'яті першого рівня).
- ***pg*** (PaGing), біт 31, — *дозвіл (pg = 1) чи заборона (pg = 0) сторінкового перетворення*. Прапор використовується при сторінковій моделі організації пам'яті.

Регістри керування

- Регістр **cr2** використовується при сторінковій організації оперативної пам'яті для реєстрації ситуації, коли поточна команда звернулася за адресою, що міститься в сторінці пам'яті, відсутньої в даний момент часу в пам'яті. У такій ситуації в мікропроцесорі виникає виняткова ситуація з номером 14, і лінійна 32-бітна адреса команди, що викликала це виключення, записується в регістр *cr2*. Маючи цю інформацію, оброблювач виключення 14 визначає потрібну сторінку, здійснює її підкачування в пам'ять і відновляє нормальну роботу програми;
- Регістр cr3 також використовується при сторінковій організації пам'яті. Це так називаний *регістр каталогу сторінок першего рівня*. Він містить 20-бітну фізичну базову адресу каталогу сторінок поточної задачі. Цей каталог містить 1024 32-бітних дескриптора, кожний з який містить адресу таблиці сторінок другого рівня. У свою чергу кожна з таблиць сторінок другого рівня містить 1024 32-бітних дескриптора, що адресують сторінкові кадри в пам'яті. Розмір сторінкового кадру — 4 Кбайт.

Регістри системних адрес

- Ці регістри ще називають *регістрами керування пам'яттю*. Вони призначені для захисту програм і даних у мультизадачном режимі роботи мікропроцесора.
- При роботі в захищеному режимі мікропроцесора адресний простір поділяється на:
- *глобальне* — загальне для всіх задач;
- *локальне* — окреме для кожної задачі.

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

- *регістра таблиці глобальних дескрипторів* **gdtr** (Global Descriptor Table Register) розмір, що має, 48 біт і утримуючого 32-бітову (біти 16-47) базову адресу глобальної дескрипторної таблиці GDT і 16-бітове (біти 0-15) значення межі, що представляє собою розмір у байтах таблиці GDT;

Регістри системних адрес

- *регістра таблиці локальних дескрипторів* **ldtr** (Local Descriptor Table Register) розмір, що має, 16 біт і утримуючого так називаний селектор дескриптора локальної дескрипторної таблиці LDT. Цей селектор є покажчиком у таблиці GDT, що і описує сегмент, що містить локальну дескрипторну таблицю LDT;
- *регістра таблиці дескрипторів переривань* **idtr** (Interrupt Descriptor Table Register) розмір, що має, 48 біт і утримуючого 32-бітову (біти 16-47) базову адресу дескрипторної таблиці переривань IDT і 16-бітове (біти 0-15) значення межі, що представляє собою розмір у байтах таблиці IDT;
- *16-бітового регістра задачі* **tr** (Task Register), що подібно регістру ldtr, містить селектор, тобто покажчик на дескриптор у таблиці GDT. Цей дескриптор описує *поточний сегмент стану задачі* (TSS — Task Segment Status). Цей сегмент створюється для кожної задачі в системі, має жорстко регламентовану структуру і містить контекст (поточний стан) задачі. Основне призначення сегментів TSS — зберігати поточний стан задачі в момент переключення на іншу задачу.

Регістри налагодження

- Це дуже цікава група регістрів, призначених для апаратного налагодження. Засоби апаратного налагодження вперше з'явилися в мікропроцесорі i486. Апаратно мікропроцесор містить вісім регістрів налагодження, але реально з них використовуються тільки 6.
- Регістри **dr0, dr1, dr2, dr3** мають розрядність 32 біт і призначені для завдання лінійних адрес чотирьох крапок переривання. Використовуваний при цьому механізм наступний: будь-яка формована поточною програмою адреса порівнюється з адресами в регістрах *dr0...dr3*, і при збігу генерується виключення налагодження з номером 1.
- Регістр **dr6** називається регістром стану налагодження. Біти цього регістра установлюються відповідно до причин, що викликали виникнення останнього виключення з номером 1.

Регістри налагодження

- Перелічимо ці біти і їхнє призначення:
- ***b0*** — якщо цей біт встановлений у 1, то останнє виключення (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 налагодження;
- ***bs*** (біт 14) — встановлюється в 1, якщо виключення 1 було викликано станом прапора *tf* = 1 у регістрі *eflags*;
- ***bt*** (біт 15) встановлюється в 1, якщо виключення 1 було викликано переключенням на задачу з установленим бітом пастки в TSS *t* = 1.
- Всі інші біти в цьому регістрі заповнюються нулями. Оброблювач виключення 1 по вмісту *dr6* повинний визначити причину, по якій відбулося виключення, і виконати необхідні дії.

Регістри налагодження

Регістр *dr7* називається регістром керування налагодженням. У ньому для кожного з чотирьох регістрів контрольних крапок налагодження маються поля, за допомогою яких можна уточнити наступні умови, при яких варто згенерувати переривання:

- *місце реєстрації контрольної крапки* — тільки в поточній задачі чи в будь-якій задачі. Ці біти займають молодші вісім біт регістра *dr7* (по двох біта на кожну контрольну крапку (фактично крапку переривання), що задається регістрами *dr0, dr1, dr2, dr3* відповідно). *Перший* біт з кожної пари — це так називаний *локальний* дозвіл; його установка говорить про те, що крапка переривання діє якщо вона знаходиться в межах адресного простору поточної задачі. *Другий* біт у кожній парі визначає *глобальний* дозвіл, що говорить про те, що дана контрольна крапка діє в межах адресних просторів усіх задач, що знаходяться в системі;
- *тип доступу*, по якому ініціюється переривання: тільки при вибірці команди, при записі чи при записі/читанні даних. Біти, що визначають подібну природу виникнення переривання, локалізуються в старшій частині даного регістра.
- Більшість із системних регістрів програмно доступні. Не усі з них знадобляться в нашому подальшому викладі, але, проте, я коротко розглянув їх для того, щоб збудити в читача інтерес до подальшого дослідження архітектури мікропроцесора.