Властивості/принципи роботи машини фон Неймана

1. Лінійний простір пам’яті. (Для оперативного зберігання інформації комп’ютер має сукупність комірок з послідовною нумерацією (адресами) 0, 1, в залежностіі від об’єму (розрядності процесора x32 = 2^32 або, приблизно, 4ГБ)
2. Збереження програми та даних у пам’яті (оперативній). (код програми та її дані знаходяться в одному адресному просторі оперативної пам’яті)
3. Мікропрограмування. (Машинна мова не є тією кінцевою. До складу процесора входить блок мікропрограмного управління. Цей блок для кожної машинної команди має набір дій-сигналів, які потрібно згенерувати для фізичного виконання необхідної машинної команди.)
4. Послідовне виконання програм. (Процесор вибирає з пам’яті команди виключно послідовно. Для зміни прямолінійного виконання програм необхідно використовувати спеціальні команди (команди умовного та безумовного переходу. Паралелізм, багатозадачність з’явились набагато пізніше)
5. Відсутність різниці між командами програм та даними. (З погляду процесора принципової різниці між даними та командами немає. Дані та команди знаходяться в одному просторі пам’яті у вигляді послідовності нулів та одиниць. Тому в програмі важливо завжди чітко розділяти простір даних і команд.)
6. Індиферентність до цільового призначення даних. (Для машини немає значення, яке логічне навантаження несуть дані, які обробляються)

Програма у фон-нейманівській ЕОМ реалізується центральним процесором (ЦП) за допомогою послідовного виконання створюючих цю програму команд. Дії, потрібні для вибірки і виконання команди, називають циклом команди. В загальному випадку цикл команди включає декілька складових (етапів):
вибірку команди;

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

Перераховані етапи виконання команди надалі називатимемо стандартним циклом команди. Відзначимо, що не всі з етапів присутні під час виконання будь-якої команди (залежить від типу команди), проте етапи вибірки, декодування, формування адреси наступної команди і виконання мають місце завжди.

Стисло охарактеризуємо кожен з вищеперелічених етапів стандартного циклу команди.

Етап вибірки команди.

Цикл будь-якої команди починається з того, що центральний процесор витягує команду з пам'яті, використовуючи адресу, що зберігається в лічильнику команд (ЛК). Двійковий код команди поміщається в регістр команди (РК) і з цієї миті стає «видимим» для процесора. Лічильник команд і регістр команд розташовані в пристрої управління. Система команд багатьох ОМ припускає декілька форматів команд, причому в різних форматах команда може займати 1, 2 або більше комірок. У цьому випадку етап вибірки команди можна вважати завершеним лише після того, як в РК буде поміщений повний код команди. Інформація про фактичну довжину команди міститься в полях коду операції і способу адресації. Зазвичай ці поля розташовують у першому слові коду команди, і для з'ясування необхідності продовження процесу вибірки необхідне попереднє декодування їх вмісту. Таке декодування може бути проведене після того, як перше слово коду команди опиниться в РК. У разі багатослівного формату команди процес вибірки продовжується аж до занесення в РК всіх слів команди.

Етап формування адреси наступної команди.

Для фон-нейманівських машин характерне розміщення сусідніх команд програми в суміжних комірках пам'яті. Якщо вибрана з пам'яті команда не порушує природного порядку виконання програми, то для обчислення адреси наступної виконуваної команди досить збільшити вміст лічильника команд на довжину поточної команди, яка подана кількістю займаних кодом команди комірок пам'яті. Довжина команди, а також те, чи здатна вона змінити природний порядок виконання команд програми, з'ясовуються в ході раніше згадуваного попереднього декодування.

Якщо вибрана з пам'яті команда здатна змінити послідовність виконання програми (команда умовного або безумовного переходу, виклику

процедури і т. п.), процес формування адреси наступної команди переноситься на етап виконання операції. В силу сказаного, в ряді ОМ даний етап циклу команди йде не за вибіркою команди, а знаходиться в кінці циклу.

Етап декодування команди.

Після вибірки команди вона повинна бути декодована, для чого ЦП розшифровує код команди, що знаходиться в РК. В результаті декодування з'ясовуються такі моменти:

- чи знаходиться в РК повний код команди або потрібне дозавантаження решти слів команди;
- які подальші дії потрібні для виконання даної команди;
- якщо команда використовує операнди, то звідки вони повинні бути узяті (номер регістра або адреса комірки основної пам'яті);
- якщо команда формує результат, то куди цей результат повинен бути направлений.

Відповіді на два перші питання дає розшифровка коду операції, результатом якої може бути унітарний код, де кожен розряд відповідає одній з 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 вказаної в команді операції.

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

За наслідками декодування проводиться підготовка електронних схем ОМ до виконання наказаних командою дій.

Етап обчислення адрес операндів.

Етап 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 вказаного в команді способу адресації. Так, у разі індексної адресації для отримання виконавчої адреси проводиться підсумовування вмісту адресної частини команди і вмісту індексного регістра.

Етап вибірки операндів.

Обчислені на попередньому етапі виконавчі адреси використовуються для зчитування операндів з пам'яті і занесення в певні регістри процесора. Наприклад, у разі арифметичної команди операнд після вибору з пам'яті може бути завантажений у вхідний регістр АЛП. Проте частіше операнди заздалегідь заносяться в спеціальні допоміжні регістри процесора, а їх пересилка на вхід АЛП відбувається на етапі виконання операції.

Етап виконання операції.

На цьому етапі в АЛП реалізується вказана в команді операція. Через відмінність суті кожної з команд ОМ, зміст цього етапу також суто індивідуальний.

Етап запису результату.

Етап запису результату присутній в циклі тих команд, які припускають занесення результату в регістр або комірку основної пам'яті. Якщо отриманий результат буде використаний в наступній команді, то він може залишатися в регістрі АЛП.

Структура сучасного комп'ютера

Персональний комп'ютер (ПК) є сукупністю двох важливих частин:

апаратного та програмного забезпечення. На сьогоднішній день найбільше розповсюдження отримали персональні комп’ютери, структурна схема яких подана на слайді.

Архітектура сучасного комп'ютера являє собою опис побудови комп’ютера і його окремих складових та визначається рядом властивостей, які забезпечують можливість функціонування програмного забезпечення, що управляє периферійним обладнанням. Програми можуть взаємодіяти з пристроями різними способами:

- через виклики функцій операційної системи;
- через виклики функцій базової системи вводу/виводу (BIOS);
- безпосередньо взаємодіючи з відомими їм пристроями.

У системному блоці розташовуються такі компоненти:

Системна плата (об'єднуюча, материнська) є основним компонентом комп'ютера. Вона, як правило, містить:

сокет (слот) процесора;

сокети (слоти) пам'яті ОЗП;

слоти шини;

ПЗП BIOS (ROM BIOS);

Набір мікросхем системної логіки містить усі схеми, які входять до складу системної плати. Він управляє центральним процесором, системною шиною процесора, кешем L2, L3, оперативною пам'яттю, шиною PCI (Peripheral Component Interconnect), та іншими системними ресурсами.

Набір мікросхем системної логіки визначає також первинні можливості та специфікації системної плати – типи підтримуваних процесорів, пам’яті, плат розширення, дисководів та ін.

Системні плати випускають у кількох варіантах, які відрізняються розмірами та форм-факторами. Форм-фактор системної плати визначає тип корпусу, в який її можна вставити.

Сучасні системні плати будують на основі чипсетів (Chipset) – набір з декількох надвеликих інтегральних схем, що реалізують усі необхідні функції зв'язку основних компонентів: МП, пам'яті і шин розширення. Майже всі сучасні чипсети є набором із двох мікросхем, що прийнято називати Northern Bridge (Північний міст) і Southern Bridge (Південний міст).

Northern Bridge відповідає за роботу з МП, системною шиною, PCI-шиною, AGP-портом, пам'яттю і кешем (які не входять до складу МП).

Southern Bridge – це фактично простий PCI – пристрій, що містить всередині себе контролер Bus Master IDE (який дозволяє пристроям, що знаходяться на такій шині, самим керувати процесом передачі даних по ній без участі процесора).

Периферія досить стабільна, тому архітектура Southern Bridge набагато менше піддається змінам, чим Northern Bridge, що дозволяє в нових чипсетах використовувати його попередні версії.

Мікропроцесор (МП) – це основний «мозковий» вузол ПК, в задачу якого входить виконання програмного коду, що знаходиться в пам'яті, і керування іншими пристроями. До складу мікропроцесора сучасного ПК крім центрального мікропроцесора – пристрою обробки CPU (Central Processing Unit) – входить математичний співпроцесор CoP (CoProcessor), який ефективно обробляє числові дані у форматі з плаваючою комою (крапкою), і невелика за обсягом швидкодіюча кэш-пам'ять, реалізована за одно- або дворівневою схемою (від англ. cache – склад, схованка; кеш-пам'ять призначена для пом'якшення наслідків, що викликані розузгодженням швидкості роботи швидкого МП і повільної оперативної пам'яті).

Мікропроцесор – це найчастіше одна надвелика інтегральна схема (НВІС), реалізована в єдиному напівпровідниковому кристалі. Ступінь інтеграції визначається розміром кристала і кількістю реалізованих у ньому транзисторів.

Деякі МП, строго кажучи, не є однокристальними – кристал центрального процесора зі співпроцесором і кілька кристалів вторинного кеша зібрані на загальному картриджі.

У багатопроцесорній системі для підвищення загальної продуктивності системи функції центрального процесора розподіляються між декількома, звичайно ідентичними мікропроцесорами, один із яких призначається головним.

Співпроцесор – це спеціалізований процесор, призначений для розвантаження центрального процесора від складних обчислень з плаваючою комою і є вбудованим у середину кристала мікропроцесора.

Кеш–пам'ять першого рівня (Level 1 – L 1) вбудована у середину кристала і працює на однаковій з ним частоті. Якщо МП використовує дворівневу модель кеш-пам'яті (L1, L2), то застосовується архітектура подвійної незалежної шини (ПНШ – Dual Independent Bus). Одна з шин МП архітектури ПНШ – локальна шина – використовується тільки для зв'язку з

кристалами вторинного кеша, які розташовані у тому ж корпусі мікросхеми або на загальному картриджі. Ця шина є локальною і у геометричному значенні – провідники мають довжину одиниць сантиметрів, що дозволяє її використовувати навіть на частоті ядра процесора.

Друга шина МП виходить на зовнішні виводи мікросхеми, вона є системною шиною МП. Ця шина працює на зовнішній частоті незалежно від внутрішньої шини.

МП керує роботою ПК, отримуючи і посилаючи керуючі сигнали, адреси пам'яті і дані від одних компонентів ПК до інших компонентів, використовуючи для цього групу сполучаючих електронних шляхів, які називаються системною шиною (СШ).

Системна шина - це електронний шлях на системній платі, що спільно використовується, до якого підключені всі керовані компоненти ПК. Коли дані передаються від одного компонента до іншого, вони переміщаються вздовж цього загального шляху до місця призначення. СШ поділяється на чотири складові частини: лінії передачі електроживлення, керуюча шина (для передачі керуючої інформації, наприклад, такої, як тактові сигнали від системного генератора тактових сигналів, сигнали переривання і т.д.), адресна шина (виконує передачу адрес комірок пам'яті і пристроїв, приєднаних до шини) і шина даних (разом із шиною адреси здійснює перенесення даних в середині ПК).

Оскільки швидкодія різних компонентів ПК (мікропроцесора, пам'яті й інших пристроїв) істотно розрізняється, в комп'ютерах на МП застосовується внутрішнє множення частоти. Розрізняють такі частоти:

Host Bus Clock – частота системної шини (зовнішня частота процесора), опорна для всіх інших частот.

CPU Clock чи Core Speed - внутрішня частота МП, на якій працює його обчислювальне ядро (центральний процесор, співпроцесор, кэш-пам'ять L1). Дозволяє підвищити граничні частоти інтегральних компонентів, для чого використовується внутрішнє множення частоти.

Основна чи оперативна пам'ять ПК призначена для оперативного обміну (збереження, запису і зчитування) інформацією (кодами і даними) між МП, зовнішньою пам'яттю і периферійними пристроями. Для побудови основної пам'яті в ПК використовують мікросхеми динамічної пам'яті (мікросхеми DRAM – пам'яті – Dynamic Random Access Memory), що мають найкраще сполучення обсягу, щільності упакування, енергоспоживання і ціни.

Мікросхеми DRAM – пам'яті в сучасному ПК установлюють на спеціальні модулі пам'яті SIMM (Single In – Line Memory Module), DIMM (Dual In – Line Memory Module), RIMM (Rambus In – Line Memory Module) у відповідні гнізда системної плати.

Важливою особливістю основної пам'яті ПК є її ієрархічний спосіб побудови, що полягає в сполученні основної пам'яті великого обсягу на мікросхемах динамічної пам'яті з відносно невеликою кеш-пам'яттю на швидкодіючих мікросхемах статичної пам'яті SRAM (Static RAM).

Кеш-пам'ять є додатковим і швидкодіючим сховищем копій блоків інформації основної пам'яті, до яких, імовірно, найближчим часом буде звернення. У сучасному комп'ютері кеш-пам'ять побудована за трирівневою схемою:

кеш L1 – кеш, вбудований в кристал мікропроцесора класу 486 і старше, працює на CPU Clock;

кеш L2 – кеш вбудований в корпус МП чи встановлений на загальному працює на CPU Clock чи на половині цієї частоти;

кеш L3 – кеш на системній платі мікропроцесора працює на Host Bus Clock.

Кеш на системній платі сучасного ПК набирається мікросхемами статичної пам'яті фіксованого обсягу, запаяних на плату без застосування додаткових модулів і роз'ємів.

CMOS Memory (Complementary Metal-Oxide-Semiconductor), RTC (Real Time Clock) – спеціальна мікросхема напівпостійної пам'яті (КМОН – пам'ять) невеликого обсягу для збереження інформації про конфігурацію ПК разом з годинником і календарем. Живлення CMOS Memory, RTC при виключеному ПК здійснюється від батарейки.

BIOS (Basic Input Output System) – ключовий елемент системної плати, призначений для енергонезалежного збереження системної інформації. BIOS, користуючись засобами, наданими чипсетом, керує всіма компонентами і ресурсами системної плати. Код BIOS зберігається в мікросхемі енергонезалежної постійної пам'яті (Read Only Memory (ROM BIOS) чи флеш-пам'яті (Flash BIOS).

Шини розширення призначені для підключення різних адаптерів чи контролерів периферійних пристроїв, що розширюють можливості ПК. Під адаптером звичайно розуміють засіб сполучення якого-небудь пристрою з шиною ПК (контролер служить тим же цілям сполучення, але при цьому мається на увазі його деяка активність, тобто здатність до самостійних дій після отримання команд від обслуговуючої його програми).

В сучасному ПК застосовують такі шини, розташовані на системній платі у виді слотів чи секцій розширення:

PCI – Peripheral Component Interconnect Local Bus;

AGP – Accelerated Graphic Port – прискорений графічний порт;

Зовнішні інтерфейси ПК (рівнобіжний, послідовний і USB-шина), як і шини розширення, дозволяють значно розширити функціональні можливості ПК. Однак, на відміну від шин розширення, зовнішні інтерфейси дозволяють підключати різні периферійні пристрої прямо, без використання додаткових адаптерів.

Зовнішні запам'ятовуючі пристрої комп'ютера (ЗЗП) – це пристрої, що дозволяють автономно зберігати інформацію для наступного її використання незалежно від стану ПК (включений чи виключений). В сучасному ПК в ЗЗП входять пристрої магнітної, оптичної, магнітооптичної та твердотільної пам'яті. ЗЗП можна розміщувати як у системному блоці комп'ютера, так і в окремому корпусі.

Блок живлення призначений для перетворення змінного електричного струму в постійний, який стабілізований у невеликих межах, а також для захисту електричних ланцюгів блока живлення і комп'ютера від впливу різних перешкод і несправностей.

Отже, це наближена структура комп’ютера, його основні компоненти, які можуть відрізнятись в залежності від конкретної архітектури, котрих є досить багато: RISC-архітектура – CISC-архітектура, з повним набором команд, зі скороченим набором команд, є поняття x86/IA-32 архітектура, x86-64 архітектура, ARM-архітектура і таке інше.

Якщо вести мову про архітектури з якими и матимемо справу – x86/IA-32 та архітектура x86-64 – це архітектури розроблені свого часу компанією INTEL та в подальшому удосконаленою компанією AMD.

Назву архітектура отримала від процесора 8086, який потім розвивався в 80186, 80286, 386, 486, Pentium 1, 2, 3, 4 та інші більш сучасні процесори, але принципи залишались тими самими. Ці процесори були орієнтовані на оперування 32 розрядними даними, мали 32 розрядні регістри і досить широко використовувались.

Коли виникла проблема необхідності на 64 розрядні дані, компанія Intel розробляла x64 архітектуру, але вона була складна у реалізації і не прижилась, а компанія AMD модифікувала архітектуру до x86-64 яка зараз широко використовується.

ПРОГРАМНА МОДЕЛЬ ПРОЦЕСОРА IA-32

2.4.1. Склад програмної моделі

Будь-яка програма, що виконується, має у своєму розпорядженні визначений набір ресурсів процесора. Ці ресурси необхідні для виконання та збереження в пам’яті команд програми, даних та інформації про поточний стан програми й процесора. Набір таких ресурсів представляє програмну модель процесора.

Програмна модель – це ресурси процесора, як доступні для використання в програмах, так і приховані від програміста (використовуються тільки процесором).

Базова програмна модель включає:

- простір адресовної пам’яті до 2^{32}–1 байт (4 Гбайт), а для відповідних технологій – до 2^{36}–1 байт (64 Гбайт);

- набір регістрів для зберігання даних загального призначення;

- набір сегментних регістрів, призначених для зберігання шести селекторів сегментів;

- набір регістрів стану та управління;

- набір регістрів пристрою розрахунків із рухомою комою (сопроцесор);

- набір регістрів цілочислового MMX-розширення;

- набір регістрів MMX-розширення з рухомою комою;

- програмний стек. (Це спеціальним чином організована область оперативної пам'яті, що дозволяє з використанням відповідних інструкцій розміщувати в ній ті чи інші дані)

Це основний набір ресурсів. Крім того, до ресурсів, які підтримуються архітектурою IA-32, необхідно віднести порти вводу/виводу, лічильники моніторингу продуктивності.

Набір регістрів.

Регістрами називають області швидкодіючої пам’яті, які розташовані всередині процесора в безпосередній близькості від його виконавчого ядра. Доступ до них здійснюється набагато швидше, ніж до комірок оперативної пам’яті. Машинні команди з операндами в регістрах виконуються максимально швидко.

Більшість регістрів мають визначене функціональне призначення.

З погляду програміста їх можна розділити на дві великі групи.

Першу групу утворюють користувацькі регістри, до яких належать:

- регістри загального призначення (EAX/AX/AH/AL, EBX/BX/BH/BL, EDX/DX/DH/DL, ECX/CX/CH/CL, EBP/BP, ESI/SI, EDI/DI, ESP/SP) використовуються для збереження даних та адрес, програміст може використовувати їх для реалізації своїх алгоритмів однак із деякими обмеженнями;

AL (8 bit) -> AX (16 bit) -> EAX (32 bit) -> REAX або R1 (64 bit)

Можна оперувати регістрами AL, AH, AX, EAX REAX

Регістри загального призначення використовуються для збереження:

– операндів логічних і арифметичних операцій;

– компонент адрес;

– покажчиків на комірки пам’яті.

Усі ці регістри доступні для зберігання операндів без особливих обмежень, хоча в деяких випадках деякі з них мають певне функціональне призначення.

Усі регістри цієї групи дають можливість звертатися до своїх “молодших” частин. Звернення здійснюється за їх іменами. Важливо зазначити, що використовувати для самостійної адресації можна тільки 16- та 8-розрядні частини цих регістрів. Старші 16 бітів для самостійної роботи не доступні.

До регістрів загального призначення належать регістри, які фізично розміщені всередині ALU, тому їх інколи називають регістрами ALU:

- EAX/AX/AH/AL (Accumulator register) – регістр-акумулятор, застосовується для збереження результатів проміжних операцій, використовується у всіх операціях введення-виведення, в арифметичних операціях, у деяких командах його використання обов’язкове;

- EBX/BX/BH/BL (Base register) – базовий регістр, застосовується для збереження базової адреси деякого об’єкта в пам’яті (єдиний, який використовується в індексній адресації), може використовуватись при розрахунках для зберігання проміжних результатів;

- ECX/CX/CH/CL (Count register) – регістр-лічильник, використовується як лічильник циклів та елементів при виконанні строкових операцій, може використовуватись у розрахунках для зберігання проміжних результатів;

- EDX/DX/DH/DL (Data register) – регістр даних так само, як і регістр EAX/AX/AH/AL призначений для зберігання проміжних даних, у деяких командах його пряме використання обов’язкове, а в деяких він використовується неявно.

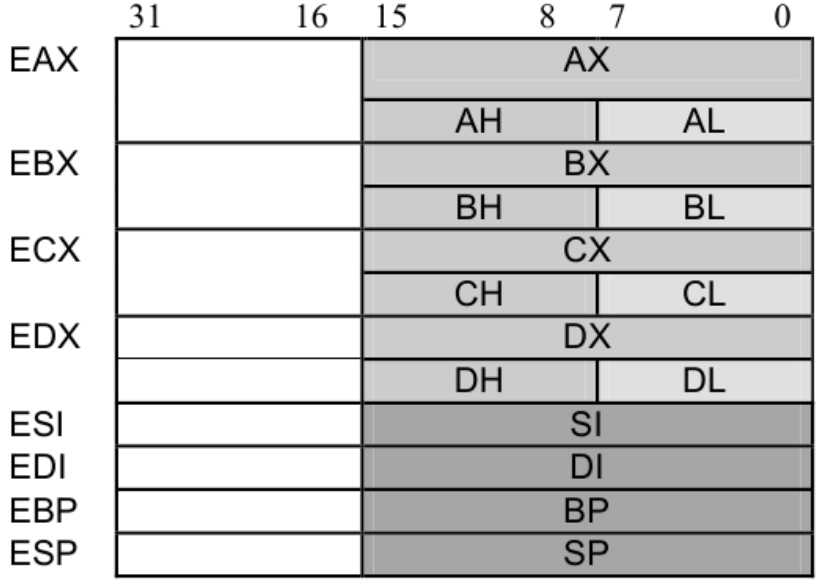
- ESI/SI (source index register) – регістр індексу джерела, містить адресу даних, які розміщуються в сегменті, що адресується регістром DS, а в ланцюгових операціях містить поточну адресу елемента в ланцюгу джерелі;

- EDI/DI (destination index register) – регістр індексу приймача, адресує дані, які розміщуються в сегменті, базова адреса якого знаходиться в регістрі ES, також може містити зміщення рядка-приймача при виконанні строкових операцій.

В архітектурі мікропроцесора на програмно-апаратному рівні підтримується така структура даних як стек. Для роботи зі стеком існують спеціальні регістри:

- ESP/SP (Stack Pointer register) – регістр покажчика стеку, який містить покажчик на верхівку стеку в поточному сегменті стеку;

- EBP/BP (Base Pointer register) – регістр покажчика бази кадру стеку призначений для організації довільного доступу до даних всередині стеку.



Регістри загального призначення цілочислового пристрою

- регістри стану й управління – це регістри, які містять інформацію про стан процесора, виконуваної програми й дозволяють змінювати цей стан (регістри прапорців EFLAGS/FLAGS та регістр покажчика команд EIP/IP).

Це досить важливі регістри, оскільки в них виводиться реакція процесора на ті чи інші події, наприклад переповнення регістра.

До складу процесора включено два регістри, які містять як інформацію про стан самого процесора, так і програми, команди якої він обробляє:

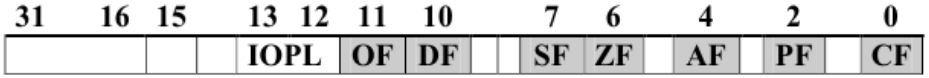
- регістр-покажчик команд EIP/IP;

- регістр прапорців (EFLAGS/FLAGS).

За допомогою цих регістрів також можна у визначених межах управляти станом процесора.

Регістр-покажчик команд EIP/IP (Instruction Pointer register) має розрядність 32(16) біти та містить зміщення наступної команди, яка буде виконуватись, відносно вмісту регістра сегмента коду CS в поточному сегменті команд. Цей регістр програмісту безпосередньо не доступний, але завантаження та зміна його значення здійснюється різними командами управління, до яких команди умовних та безумовних переходів, виклику процедур та повернення з процедур. Виникнення переривань також призводить до модифікації регістра EIP/IP. Вміст цього регістра можна прочитати виконавши команду call, після чого переглянути вміст стеку – воно буде вказувати на наступну команду.

Розрядність регістра прапорців EFLAGS/FLAGS (flag register) дорівнює 32(16) біт. Окремі розряди цього регістра мають своє функціональне призначення й називаються прапорцями. Молодша частина регістра EFLAGS повністю відповідає регістру FLAGS процесора i8086. На рис. показано структуру регістра EFLAGS.



Структура регістра EFLAGS

Прапорці регістра EFLAGS/FLAGS за функціональним призначенням поділяються на три групи.

До першої групи прапорців входять 8 прапорців стану. Ці прапорці можуть змінюватися після виконання машинних команд. Прапорці стану регістра EFLAGS відбивають особливості результату виконання арифметичних або логічних операцій. Це дає можливість аналізувати стан обчислювального процесу та реагувати на нього за допомогою команд умовних переходів та викликів підпрограм.

Розглянемо призначення прапорців.

Мнемоніка	Назва прапорця	Номер біта	Вміст та призначення
CF	Прапорець переносу (*Carry Flag*)	0	Встановлюється в 1, якщо арифметична операція зробила перенесення зі старшого біта результату. Старшим є 7, 15 або 31-й біт залежно від розмірності операнда; 0 – перенесення не було
PF	Прапорець паритету (*Parity Flag*)	2	Встановлюється в 1, якщо вісім молодших розрядів результату містять парну кількість одиниць (цей прапорець тільки для 8 молодших розрядів операндів будь-якого розміру). 0 – вісім молодших розрядів результату містять непарну кількість одиниць
AF	Допоміжний прапорець переносу (*Auxiliary Flag*)	4	Тільки для команд, працюючих з BCD-числами. Фіксує факт позики з молодшої тетради результату: 1 – в результаті операції додавання було зроблено перенесення з розряду 3 в старший розряд або при відніманні була позика в розряд 3 молодших тетради зі значення в старшій тетраді; 0 – перенесень і позик в(з) 3 розряд(а) молодшої тетради результату не було
ZF	Прапорець нуля (*Zero Flag*)	6	Встановлюється в 1, якщо результат операції дорівнює 0, і встановлюється в 0, якщо результат не нульовий
SF	Прапорець знака (*Sign Flag*)	7	Відбиває стан старшого біта результату (біти 7, 15 або 31), тобто знак результату операції, при від'ємному результаті SF = 1

Закінчення табл. 2.1

Мнемоніка	Назва прапорця	Номер біта	Вміст та призначення
OF	Прапорець переповнення (*Overflow Flag*)	11	Фіксує факт втрати значущого біта при арифметичних операціях, тобто ситуацію переповнення (вихід результату арифметичної операції за межі припустимого діапазону значень), OF = 1, якщо внаслідок операції виникає перенос у старший знаковий біт результату або позика із старшого знакового біта результату, OF=0, якщо внаслідок операції не виникло переносу у старший знаковий біт результату або позиці зі старшого знакового біта результату
IOPL	Рівень привілейованості вводу-виводу (*Input/ Output privilege level*)	12, 13	Використовується в захищеному режимі роботи процесора для контролю доступу до команд вводу-виводу залежно від привілейованості команд

Прапорці стану встановлюються після виконання операцій, результатами яких є беззнакові цілі числа, цілі числа зі знаком та упаковані (BCD) цілі числа. Якщо результатом операції є беззнакове ціле число, то встановлення прапорця переносу CF в 1 (перенос або позика) свідчить про вихід за межі припустимого діапазону. Якщо результатом операції є ціле число зі знаком (двійкове доповнення числа), про це свідчить встановлення прапорця переповнення OF в 1.

До другої групи прапорців (група прапорців управління) регістра EFLAGS/FLAGS входить лише один прапорець DF (Direction Flag) – прапорець напрямку, який використовується командами обробки рядків. Значення DF визначає напрямок поелементної обробки в цих операціях: від початку рядка до кінця чи навпаки. Якщо DF = 0, рядок обробляється в прямому напрямку, від менших адрес до старших, якщо DF = 1, обробка ведеться у зворотному напрямку.

До третьої групи прапорців регістра EFLAGS/FLAGS входять 8 системних прапорців, які керують введенням-виведенням, маскованим перериванням, відлагоджуванням, перемиканням між задачами та режимом віртуального процесора i8086. Прикладним програмам не рекомендується модифікувати без необхідності ці прапорці, через те, що у більшості випадків це спричиняє зупинку роботи програми.

– сегментні регістри CS, DS, SS, ES, FS, GS використовуються для збереження адрес сегментів у пам'яті;

Процесор Intel апартно підтримує сегментну організацію програми.

Це означає, що будь-яка програма складається з трьох сегментів: коду, даних та стеку. Логічно машинні команди в архітектурі IA-32 побудовані так, що при виборі кожної команди для доступу до даних програми або стеку неявно використовується інформація, яка розміщена у визначених сегментних регістрах. Залежно від режиму роботи процесора за їх вмістом визначаються адреси пам'яті, з яких починаються відповідні сегменти. У програмній моделі IA-32 є шість сегментних регістрів CS, SS, DS, ES, GS та FS, які призначені для доступу до чотирьох типів сегментів.

Сегмент коду містить команди програми. Для доступу до цього сегмента служить регістр сегмента коду CS (Code segment register). Він містить

адресу сегмента з машинними командами, до якого має доступ процесор (ці команди завантажуються в конвеєр процесора).

Сегмент даних містить дані, що обробляються програмою. Для доступу до цього сегмента служить регістр сегмента даних DS (Data segment register), у якому зберігається адреса сегмента даних поточної програми.

Сегмент стеку являє собою область пам'яті, яка називається стеком. Роботу зі стеком процесор організовує за таким принципом:

останній записаний у цю область пам'яті елемент вибирається першим. Для доступу до цієї області призначений регістр сегмента стеку SS (Stack segment register), який містить адресу сегмента стеку.

Додатковий сегмент даних. Неявно алгоритми виконання більшості машинних команд передбачають, що дані, які ними обробляються, розташовані в одному сегменті даних, адреса якого розміщена в регістрі сегмента даних DS. Якщо програмі недостатньо одного сегмента даних, то вона має можливість задіяти ще три додаткових сегменти даних. Але на відміну від основного сегмента даних, адреса якого розміщується в DS, при використанні додаткових сегментів даних їх адреси потрібно вказувати явно за допомогою спеціальних префіксів перевизначення сегментів у команді. Адреси додаткових сегментів повинні міститися в регістрах додаткового сегмента даних ES, GS, FS (Extension Data Segment register).

– регістри сопроцесора st(0), st(1), st(2), st(3), st(4), st(5), st(6), st(7) призначені для написання програм, що використовують тип даних з рухомою комою; Розміром 80 біт

Після розробки процесорів Pentium були додані регістри MMX-розширення:

– цілочислові регістри MMX-розширення mmx0, mmx1, mmx2, mmx3, mmx4, mmx5, mmx6, mmx7;

– регістри MMX-розширення з рухомою комою xmm0, xmm1, xmm2, xmm3, xmm4, xmm5, xmm6, xmm7;

На сучасних процесорах математичний співпроцесор поєднують на тому ж кристалі із процесором, тому такий поділ є вже більше логічним.

До другої групи входять системні регістри, тобто регістри призначені для підтримання різних режимів роботи, сервісних функцій, а також регістри, які є специфічними для різних моделей процесорів.

До їх складу входять:

- регістри управління CR0…CR4 визначають режими роботи процесора та характеристики поточної виконуваної задачі;

- регістри управління пам'яттю GDTR, IDTR, LDTR, TR використовуються в захищеному режимі роботи процесора для локалізації керуючих структур цього режиму;

- регістри відлагодження DR0…DR7 призначені для моніторингу та управління різними аспектами відлагодження;

- регістри типів областей пам'яті MTRR використовуються для апаратного управління кешуванням із метою надання відповідних властивостей областям пам'яті;

- машинно-залежні регістри MSR використовуються для управління процесором, контролю за його продуктивністю, отримання інформації про помилки.