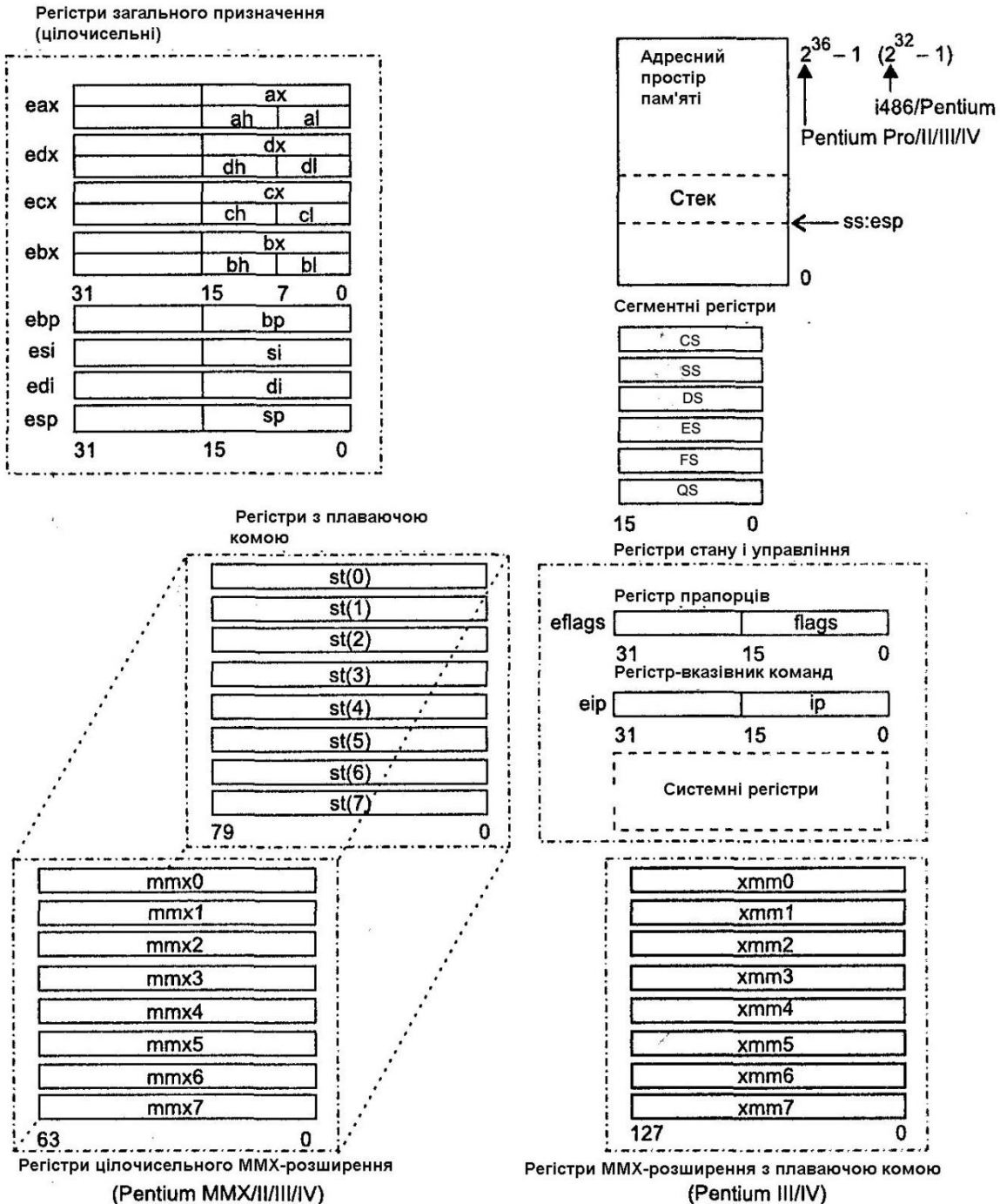
Програмна модель процесора архітектури IA-32

- **Програмна модель мікропроцесора** це незалежна від конструктивної реалізації сукупність його регістрів та оперативна пам'ять до якої мають доступ прикладні програми.
- Відповідно включає:
- - регістри;
- - ділянки памяті (сегменти).

Програмна модель процесора архітектури IA-32



- **Регістр МП**
  - надшвидка оперативна пам'ять всередині процесора, призначена для зберігання проміжних результатів обчислення, або розміщення даних необхідних для роботи процесора.

- Регістри МП можна розділити на дві великі групи:
  - **користувацькі регістри**;
  - **системні регістри.**

Користувацькі регістри

- 1. Регістри загального призначення (цілочисельні).
- 2. Сегментні регістри.
- 3. Регістри стану та управління.
- 4. Регістри математичного співпроцесора та мультимедійного розширення (MMX).
- 5. Регістри потокового розширення процесора (SSE)

Регістри загального призначення (РЗП)

- Структура

Кожен з регістрів іменується певною буквою до якої долучаються префікс та суфікс, що уточнюють його розмір.

Префікси назви

R-64 біта (тільки 64-бітна архітектура)
E-32 біта

Суфікси

X – не менше 16 біт
H – старший байт 16 бітного регістра (X)
L –молодший байт 16 бітного регістра (X)

Цілочисельні регістри

- **EAX** - **акумулятор**, операнд - джерело або приймач результату (багато команд МП обумовлюють розміщення операнду в даному регістрі) ;
- **EBX** - покажчик на дані в сегменті DS (**регіст бази**) (base);
- **ECX** - **лічильник** для **ланцюгових** команд (наприклад, MOVS) і **циклічних** (із префіксом REP і т.п.) інструкцій (counter);
- **EDX** - адреса порту введення-виводу для інструкцій IN/INS, OUT/OUTS (data);

- **ESI** - покажчик на операнд-джерело у сегменті DS для ланцюгових інструкцій (source) ;
- **EDI** - покажчик на операнд-джерело у сегменті ES для ланцюгових інструкцій (destination);
- **ESP** - містить покажчик на вершину стека і його не варто використовувати для інших цілей (ctec point).
- **EBP** - покажчик на дані в сегменті SS (регістр кадра стеку) (base point).

- В **64 бітному** режимі доступні додаткові цілочисельні регістри **R8-R15**

Сегментні регістри

- **CS** сегментний регістр коду. Він містить адресу сегмента з машинними командами, до якого має доступ мікропроцесор (тобто ці команди завантажуються в конвеєр мікропроцесора).

- **DS** - сегментний регістр даних, що зберігає адресу сегмента даних поточної програми.

- **SS** Сегмент стека. Цей сегмент являє собою область пам'яті, звану стеком.
- Роботу зі стеком мікропроцесор організує за наступним принципом: останній записаний в цю область елемент вибирається першим.

- **ES, GS, FS** Додаткові сегменти даних.
- Якщо програмі недостатньо одного сегмента даних, то вона має можливість використовувати ще три додаткових сегмента даних. Але на відміну від основного сегмента даних, адреса якого міститься в сегментному регістрі DS, при використанні додаткових сегментів даних їх адреси потрібно вказувати явно за допомогою спеціальних префіксів перевизначення сегментів у команді.

Регістри стану та керування

- **EIP Покажчик команд** містить зсув (відносна адреса) наступної команди, що підлягає виконанню. Відносна адреса відраховується від початку (або базової адреси) сегмента задачі, що виконується. Покажчик команд безпосередньо недоступний програмістові, але він керується явно командами керування потоком, перериваннями і виключеннями (JMP, CALL, RET, IRET, команди умовного переходу).

- **EFLAGS Регістр системних прапорів** містить групу прапорів стану, керування і системних прапорів. Значущими є окремі біти регістру які мають індивідуальні назви. Невизначені біти зарезервовані, тобто на даний момент вони не мають значення.
- Далі термін "установлений" означає значення 1, а термін "скинуте" - значення 0

Регістр системних прапорів

- **CF - прапор переносу (Carry Flag**). Установлено, якщо операція привела до переносу зі старшого біта при додаванні або до займу в старший біт при відніманні, інакше скинутий. Для беззнакових операцій прапор сигналізує про переповнення.

- **PF - прапор парності (Parity Flag**). Установлено, якщо молодші вісім біт операнда містять парне число одиниць (перевірка на парність) інакше скинутий..

- **AF - прапор допоміжного переносу (Adjust Flag)**. Використовується для спрощення додавання і вирахування упакованных двійково-десяткових чисел.

- **ZF - прапор нуля (Zero Flag).** Установлено, якщо всі біти результату дорівнюють нулеві, інакше скинутий.

- **SF - прапор знака (Sign Flag**). Установлено, якщо встановлено старший біт результату, інакше він скинутий.

- **OF - прапор переповнення (Overflow Flag**). Прапор установлений, якщо операція привела до переносу (займу) у знаковий (самий старший) біт результату, але не привела до переносу (займу) із самого старшого біта, або навпаки. Для операцій над числами зі знаком сигналізує про переповнення.

- **DF - прапор напрямку (Direction Flag)** керує поводженням ланцюгових інструкцій (MOVS, CMPS, SCAS, LODS, STOS).

Режими роботи процесора

Режим работы		ОС (бит)	Приложения надо откомпилировать заново?	Типичный размер адрес/операнд		Новые регистры доступны?
Long Mode	64-bit Mode	64	да	64	32	да
	Compatibility Mode		нет	32		нет
				16	16	
Legacy Mode	Protected Mode	32	нет	32	32	нет
				16	16	
	Virtual-8086 Mode			16	16	
	Real Mode	16				

- **Реальний режим (Real Mode)** У реальному режимі мікропроцесор працює як дуже швидкий 8086 з можливістю використання 32-бітних розширень.
- **Режим виртуального 8086 (Virtual 8086 Mode)** Для деяких задач може емулювання керування пам'яттю як у процесора 8086. При цьому задачі віртуального МП 8086 ізольовані і захищені, як від один одного, так і від звичайних задач захищеного режиму.
- **Захищений режим (Protection Mode)** -основний режим роботи мікропроцесора, який підтримує виртуальний адресний простір процесів, захист памяті та багатозадачність.
- **Long Compatibility mode** режим роботи 64 бітних процесорів повністю сумісний із захищеним режимом.
- **Long 64 bit mode** режим роботи 64 бітних процесорів з доступними розширеннями архітектури AMD64 (EM64T). Тільки 64-бітні операційні системи.

Моделі памяті

- Модель пам'яті – способи розміщення сегментів програми та їх розмір в оперативній пам'яті.
  - **В 16- бітному режимі:**
- **Tiny (крихітна)** Усі чотири сегментних регістри (CS, DS, SS і ES) установлюються на той самий адрес, що дає загальний розмір коду, даних і стека, рівний 64К. Використовуються винятково ближні покажчики (16 біт). Програми з крихітною моделлю пам'яті можна перетворити до формату.COM
- **Small (мала)** сегменти коду і даних розташований окремо і не перекриваються, що дозволяє мати 64К коду програми і 64К даних і стека. Використовуються тільки ближні покажчики.
- **Medium (середня)** Для коду, але не для даних використовуються далекі покажчики (32- біт). У результаті дані плюс стек обмежений розміром 64К, а код може займати до 1Мб.
- **Compact (компактна)** Ситуація, протилежна щодо моделі Medium: дальні покажчики використовуються для даних, але не для коду. Код тут обмежений 64К, а граничний розмір даних – 1 Мб.
- **Large (велика)** Далекі покажчики використовуються як для коду, так і для даних, що дає граничний розмір 1 Мб для обох.

В 32 бітному режимі:

Модель памяті	Модель коду	Модель даних	Вказівник по замовчуванню для команд	Вказівник по замовчуванню для даних
Flat	small	small	Near	**Near**
Small	small	small	Near	**Near**
Medium	Big	Small	Far	**Near**
Compact	Small	Big	Near	**Far**
Large	**Big**	**Big**	**Far**	**Far**

Small - 4GB.

Big 2**48-1

Near – 32 біти.

Far - 48 біти (16-сегментний регістр 32-біти зміщення).

Структура програми

- **Каркас програми для Windows**

- .386 ;*директива на використання команд процесора 386*
- .MODEL flat, STDCALL ; *модель памяті та правило виклику процедур*
- *OPTIONS CASEMAP:NONE ; чутливість до регістру*
- .DATA ; *директива що задає секцію обявлення ініціалізованих змінних*
- .DATA? ; дані без ініціалізації
- .CONST ; задання констант
- .CODE ; код програми
- start:
- <Ваш код>
- .....
- end <мітка> ;start
-

Компіляція (compil.bat)

1. C:\masm32\BIN\ML.EXE /c /coff lab01.asm
- ; без лінка; формат об’єктного файлу

lab01.asm lab01.obj

2. C:\masm32\BIN\LINK.EXE /subsystem:console lab01.obj
- ; windows

lab01obj lab01.exe

Синтаксис мови

- Асемблер визначає синтаксичні конструкції чотирьох типів:
- **команди**, або інструкції, що представляють собою символічні аналоги машинних команд. У процесі трансляції інструкції ассемблера перетворюються у відповідні команди системи команд мікропроцесора;
- **макрокоманди** — оформлювані певним чином пропозиції програми, що заміщаються під час трансляції іншими пропозиціями; (Починаються з .)
- **директиви**, що є вказівкою трансляторові асемблера на виконання деяких дій. У директив немає аналогів у машинному представленні;
- **рядки коментарів**, що містять будь-які символи, у тому числі і букви російського алфавіту. Коментарі ігноруються транслятором. Починається з **;**

Загальний формат команди:

- **[мітка:][префікс] КОП [операнд 1,] [операнд 2]**
- **[префікс] –** задає специфічний спосіб виконання команди
- Приклад:

```
MI01:  MOV EAX, 120
```

- **Операнди –** частина макрокоманди чи директиви над якими виконуються дії.
- В залежності від розміщення операндів Assembler розрізняє їхні наступні типи:

- **регістрові операнди –** обробляються дані, які містяться в регістрі.
- **безпосередні операнди –** задають розміщення операнда у вигляді його адреси в оперативній пам'яті
- **переміщувальні операнди** - це є символьні імена, які задають адреси розміщення у пам'яті команд чи інструкцій
- **базові та індексні операнди** - використовуються для розміщення індексної адресації
- **структурні операнди –** це є операнди структурного запису
- **неявні операнди** - команда визначає сама
- **лічильник адреси** — специфічний вид операнда, позначається знаком $. транслятор замінює його поточним значення лічильника адреси.

Цілі типи даних

- **Оголошення:** [ім’я] Dn значення
- Dn –
- **DB** – BYTE 8
- **DW** – WORD 16
- **DD** – DWORD 32
- **DQ** – QWORD 64
- **DT** – TBYTE 80
- Приклади:
- A1 DB ?
- Str1 DB ‘ACCEMLER’
- A2 DB 32
- A3 DB 20H
- A4 DW 01FFH
- A5 DW 0001110B
- A6 DD 10,11,12,13,14,15
- Str2 DB ‘MY’,1,’DOC’
- STR1 BYTE "THIS FIRST STRING1",0
- STR2 BYTE 20 DUP (0),0
- STR3 BYTE "0123456789",0
- ENDL BYTE 13,10,0

Фундаментальні типи

Фундаментальний тип	Цілий тип	Діапазон
байт (byte)	символ зі знаком (signed char)	-128...+127
	символ без знака (unsigned char)	0...255
слово (word)	коротке зі знаком (signed short)	-32768...+32767
	коротке без знака (unsigned short)	0...65535
подвійне слово (doubleword)	ціле зі знаком (signed int)	-2147483648...+2147483647
	ціле без знака (unsigned int)	0...4294967295