**КПІ ім. Ігоря Сікорського**

**Інститут прикладного системного аналізу**

**Кафедра Системного проектування**

Лабораторна рoбота №5

«Дослідження адресації команд, стеку, віртуальної адресації»

Виконав:

Студент групи ДА-92

ННК «ІПСА»

Насікан Дмитро Юрійович

Варіант № 11

Київ – 2021 рік

**ЗАВДАННЯ**

1. Перерахувати способи адресації команд і пояснити їх особливості.
2. Скласти схеми обчислення адреси наступної команди для різних режимів адресації команд.
3. Пояснити призначення регістрів, використовуваних для адресації команд.
4. Пояснити використання стеку підчас виклику функцій.
5. Пояснити принципи організації стека. Визначити склад регістрів і пояснити призначення засобів адресації стека.
6. Оформити програму обробки масиву з роботи №4 у вигляді головної програми і процедури. Зафіксувати зміни вмісту регістрів і стека при виклику підпрограми і при поверненні в основну програму. Пояснити ці зміни.
7. Описати засоби віртуальної адресації пам’яті х86 в 32-розрядному захищеному режимі. Скласти схеми обчислення адрес.

**ХІД РОБОТИ**

1. Перерахуємо способи адресації команд і пояснимо їх особливості та складемо схеми обчислення адреси наступної команди для різних режимів адресації команд:

* Природня – до адреси теперішньої команди, сегментна частина якої зберігається у регістрі CS (code segment), а зміщення – у регістрі IP (instruction pointer) додається довжина теперішньої команди.
* Відносна адресація – адреса наступної команди обчислюється як адреса попередньої команди + зміщення, яке вказується безпосередньо як операнда поточної інструкції (може бути як додатнім, так і від’ємним). Зазвичай використовується командами умовних та безумовних переходів.
* Стекова адресація – використовується при переходах до підпрограм.

1. Пояснимо призначення регістрів, використовуваних для адресації команд:

В архітектурі x86 є два регістри, у яких зберігається логічна адреса поточної команди:

CS (Code segment) – містить сегмент поточної команди.

IP (Instruction pointer) – містить зміщення поточної команди відносно сегмена, що зберігається у CS.

Змінити значення у цих регістрах в ручному режимі неможливо. Це робиться процесором автоматично.

1. Пояснимо використання стеку підчас виклику функцій:

Підчас виклику функції (команда call), у стек записуються адреса наступної інструкції, що повинна бути виконана після виконання функції. (При цьому IP регістр тепер вказує на початок функції)

Далі, у стек записуються усі аргументи, що передаються у функцію та значення рестрів, які потрібно зберегти та виконуються необхідні операції.

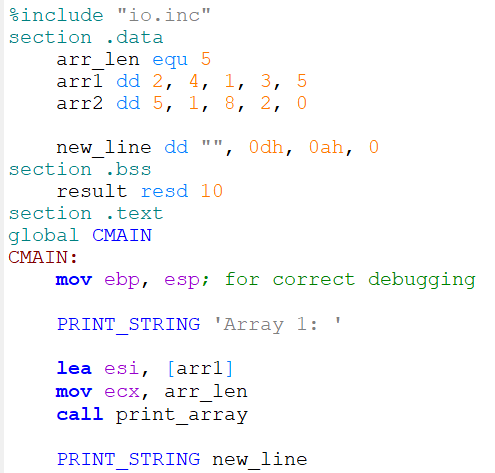
Після закінчення виконання функції, місце у стеку звільняється, та з нього дістається адреса наступної команди (за допомогою ret), що поміщається у IP регістр.

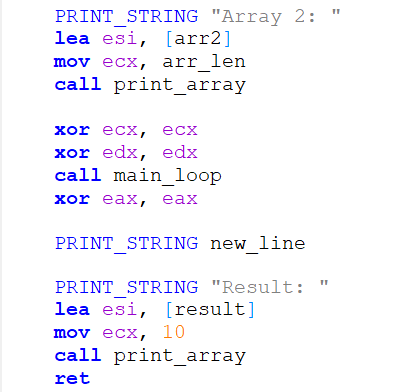
1. Пояснимо принципи організації стека. Визначимо склад регістрів і пояснимо призначення засобів адресації стека:

Стек являє собою безперервну область пам'яті, особливість якої полягає в тому, що в нього заносяться та отримуються дані за принципом "першим зайшов - останнім вийшов". Дані заносяться в стек за допомогою інструкції PUSH, а отримуються за інструкцією POP.

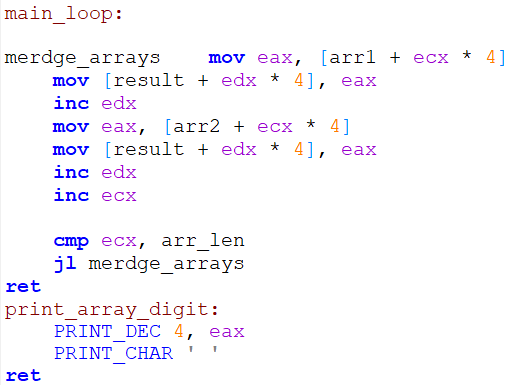
Положення стека в ОЗУ визначається вмістом регістрів SS (сегмент стека) і SP (покажчик стека). Регістр SS зберігає адресу поточного сегмента стека, а регістр SP вказує на вершину стека, тобто містить зміщення вершини стека в сегменті стеку. При кожному зверненні до стека пересилається (отримується або записується) одне слово - звичайне або подвійне. При цьому зміст SP модифікується автоматично: при запису в стек воно зменшується на 2 або 4, після чого дані записуються в сегмент, вказаний регістром SS, зі зміщенням, обумовленим новим значенням SP. При читанні з стека дані зчитуються з пам'яті за адресою SS:SP, після чого вміст SP збільшується на 2 або 4.

1. Оформимо програму обробки масиву з роботи №4 у вигляді головної програми і процедури. Зафіксуємо зміни вмісту регістрів і стека при виклику підпрограми і при поверненні в основну програму. Пояснимо ці зміни:

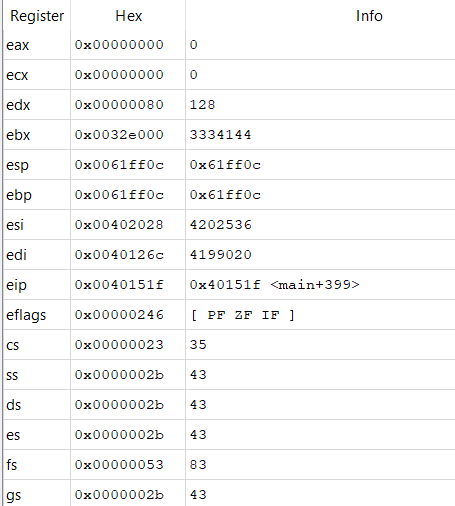




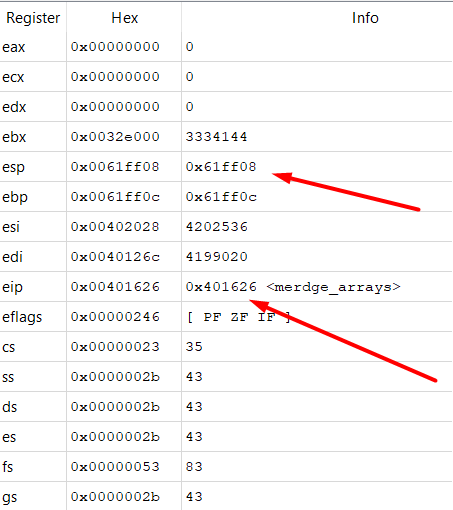
Оформимо код злиття двох масивів у вигляді процедури:



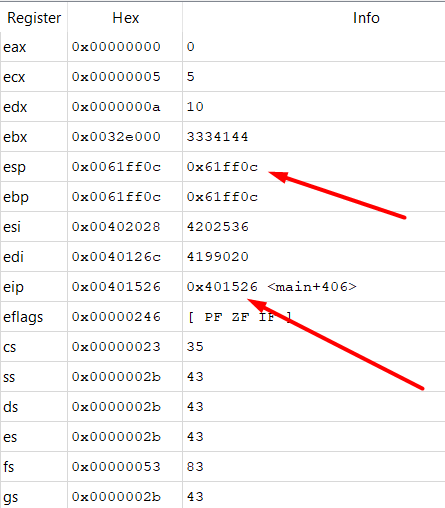
За крок до виклику процедури регістри виглядають наступним чином:



Стан регістрів одразу після виклику підпрограми:



Як бачимо, регістр eip тепер вказує на адресу початку процедури main\_loop, а в стек була завантажена адреса наступної команди, яка повинна буде виконатися після закінчення процедури.



Як бачимо, після виконання процедури, управління повертається до головної програми, про що свідчить адреса в регістрі eip, а стек був очищений.

1. Опишемо засоби віртуальної адресації пам’яті х86 в 32-розрядному захищеному режимі та складемо схеми обчислення адрес:

Адреса пам'яті в 32-розрядному процесорі є також 32-розрядною. Це означає, що адресний простір для такого процесора дорівнює 4 Гб (2^32 байт).

У захищеному режимі, також як і в реальному, існують поняття логічної і фізичної адреси. Логічна адреса в захищеному режимі (іноді використовується термін "віртуальна адреса") складається з двох 16-розрядних компонент - селектора і зсуву. Селектор записується в ті ж сегментні регістри, що і сегментна адреса в реальному режимі. Однак перетворення логічної адреси в фізичну виконується не простим складанням із зсувом, а за допомогою спеціальних таблиць перетворення адрес GDT i LDT.

Таблиця дескрипторів - це просто таблиця перетворення адрес, що містить базові 24-розрядні фізичні адреси сегментів і деяку іншу інформацію. Тобто кожен елемент таблиці дескрипторів (дескриптор) містить 24-розрядний базову адресу сегмента і іншу інформацію, що описує сегмент.

Таблиця GDT - єдина в системі. Зазвичай в ній знаходяться описи сегментів операційної системи. Таблиць LDT може бути багато. Ці таблиці містять описи сегментів програм, що працюють під управлінням операційної системи, тобто окремих завдань. У кожен даний момент часу процесор може використовувати тільки одну таблицю LDT.

**ВИСНОВКИ**

У ході даної лабораторної роботи мною було досліджені способи адресації команд та стеку, складені схеми обчислення адреси наступної команди як при переходу до наступної команди, так і при передачі управління до підпрограм. Також, було модифіковано програму з лабораторної роботи 4 та зафіксовано вміст регістрів при передачі управління у підпрограми.