**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СИКОРСЬКОГО»**

**КАФЕДРА КОНСТРУЮВАННЯ ЕЛЕКТРОННО-ОБЧИСЛЮВАЛЬНОЇ АПАРАТУРИ**

**ЗВІТ**

з лабораторної робіти № 2

по курсу «Обчислювальні та МП засоби в РЕА-2»

Виконав:

студент гр. ДК-82

Дмитрук О.О.

Перевірив:

ст. викладач

Бондаренко Н.О.

Київ – 2021

**Завдання**

1. Реалізувати ф-ю копіювання рядку в зворотньому напрямку.
2. Реалізувати ф-ю обчислення суми 32-бітних без знакових елементів масиву. Результат 64-бітне беззнакове число.
3. Реалізувати ф-ю обчислення 2-х знакових 32-бітних чисел. Результат 32-бітне число.

**Теоретична частина**

Розглянемо теорію, необхідну для виконання лабораторної роботи.

Так як ми будемо використовувати процедури, то для правильного використання процедур, потрібно дотримувати таких правил як:

1. Вхідні параметри передаються в регістрах R0-R3. Якщо ж обсяг переданих даних більше 4 \* 32 біт (важливо не кількість переданих, а їх загальний розмір), то вони передаються через R0-R3 + стек.
2. Значення, що повертається, зазвичай зберігається в R0, а якщо це значення займає 2\*32 біта, то в R0-R1.
3. Регістри R0-R3 можуть змінюватися в функції, тоді як вміст R4-R11, R12 і LR слід зберігати при вході в підпрограму і відновлювати при виході з неї. Зазвичай для цього використовують стек.
4. Значення SP має бути вирівняно за подвійним словом
5. Якщо к-сть вхідних параметрів більша за 4, в такому випадку, виконається наступне:

Завантажиться вміст параметрів в регістри, тобто 4 максимум за 1 ітерацію. Після завантажиться в стек за допомогою команди PUSH вмісти регістрів R0-R3 (якщо у нас 4 невміщувані параметри) , після завантажиться наступні параметри в регістри R0-R3.

**Виконання роботи**

Далі буде придставленна реалізація ф-ї копіювання рядку в зворотньому напрямку.

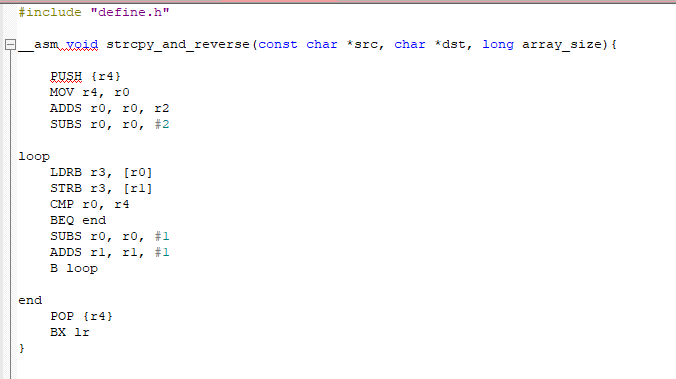


Рис.1. реалізація ф-ї копіювання рядку в зворотньому напрямку.

В якості аргументів передаються адреси 2 масивів, та їхній розмір. Тобто в регістр R0 поміститься адреса початку масиву джерела, з якого будуть братись дані. В регістр R1 поміститься адреса початку масиву приймача, в який будуть записуватись дані. В рег. R2 запишеться розмір.

Функція працює наступним чином:

Спочатку вона виконує запис в стек вмісту R4, щоб зберегти його вміст. Далі виконується обчислення адрес таким чином, щоб в R0 містилась адреса кінцевого елементу 1 масиву, це необхідно для реалізації реверсивного копіювання.

Далі переходимо в блок loop, в якому і виконується копіювання. Завантажується в регістр R3 вміст масиву Source починаючи з кінця, після завантажується вміст R3 по адресі вмісту регістру R1, тобто в масив Destination, починаючи спочатку. Потім виконується перехід на інший елемент масиву , в масиві Source це декрементування, а в Destination , інкрементування. Виконуємо даний цикл, поки вміст рег. R0 та R4 не будуть рівні (R4 містить адресу початку масиву Source)

Далі виконуємо команду Pop , щоб витягнути збережені дані зі стеку , і виконуємо перехід BX lr, щоб повернутись до викликаної команди.

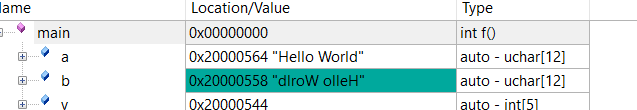


Рис.2. Результат виконання ф-ї

Далі буде представлена реалізація ф-ї обчислення суми 32-бітних без знакових елементів масиву. Результат 64-бітне беззнакове число.

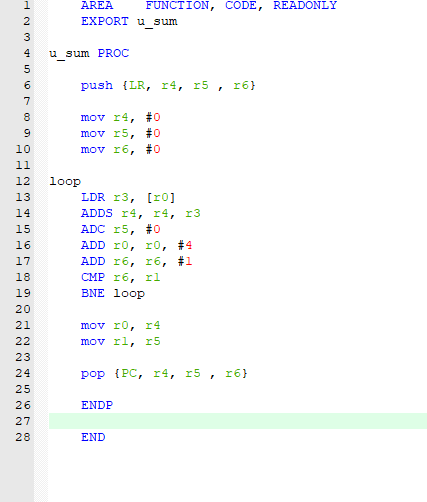


Рис.3. Реалізація ф-ї обчислення суми 32-бітних без знакових елементів масиву. Результат 64-бітне беззнакове число.

В якості параметрів передається адреса початку масиву в регістр R0, та к-сть елементів масиву в R1. Результат поміщаємо в регістр R0 та R1. Ф-я працює наступним чином:

1. Зберігаємо вміст регістрів LR, R4,R5,R6
2. Обнуляємо регістри R4,R5,R6
3. Виконуємо блок loop, в якому обчислюється додавання. Регістр R4 виступає в ролі регістру акамулятору. Тобто він постійно буде накопичуватись значеннями масиву. Після команди ADDS (додавання з врахуванням знаку, необхідно щоб флажки встановлювались) виконуємо додавання з врахуванням флажка carry, Необхідно щоб виконати перенос у випадку переповнення R4. Далі виконується перехід на наступний елемент масиву, інкрементування counter-у, і перевірка чи пройшли весь масив.
4. У випадку проходження всього масиву, виконується запис результату в регістри R0 та R1. В R0 записується молодші 32 біти результату, в R1 , старші 32 біти.

Результат виконання ф-ї:







Рис.4. Результат виконання ф-ї u\_sum

Далі буде придставлена реалізація ф-ї обчислення 2-х знакових 32-бітних чисел. Результат 32-бітне число.

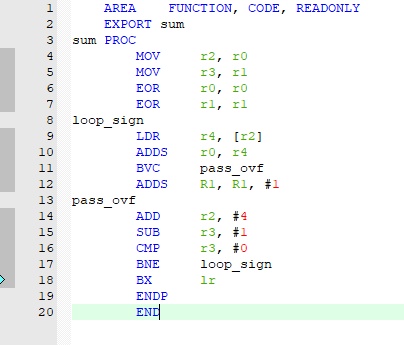
****

Рис.5. реалізація ф-ї обчислення 2-х знакових 32-бітних чисел. Результат 32-бітне число.

Параметри передаються такі ж як і в попередній ф-ї. Ф-я працює наступним чином:

1. Виконуємо запис в регістри R2 та R3 вміст регістрів R0 ,R1 відповідно. Після обнуляємо R0 та R1. Результат будемо зразу ж записувать в них.

2. Виконуємо блок loop\_sign, в якому виконується додавання знакових чисел. Далі у випадку успішного додавання, без переповнень , тобто без встановлення флажку ‘V’, виконуємо бранч на мітку pass\_ovf, де виконуємо переход на іншу строку і декремент лічильнику циклу.

Важливо зауважити, що у випадку переповнення, в регістр R1 запишеться 1, що оповістить нас, що відбулось переповнення. Також потрібно памятати що флаг Carry відповідає за перенесення та за займ , при відніманні.





Рис.6. Результат знакового додавання

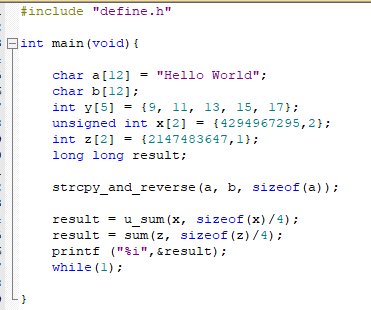


Рис.7. Головна ф-я main