Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра інформаційної безпеки та комп’ютерної інженерії

**Звіт**

**з лабораторної роботи № 2**

**на тему: «Підпрограми та стек»**

**з дисципліни: «Архітектура комп’ютера»**

ПЕРЕВІРИВ: ВИКОНАВ:

доцент студент групи

Бабенко В. Г. Кулинич О. М.

Черкаси 2016

Тема: Підпрограми та стек

Мета: Вивчення роботи підпрограм з використанням умовної адресації при вирішенні задач пов’язаних з обробкою масивів.

**Завдання**

**Варіант 3 (13)**

1. Розробити програму що знаходить різницю сум парних і непарних елементів масиву.
2. Скласти граф-схему алгоритму вирішення задачі
3. Розподілити пам’ять ОЗП.
4. Записати програму в вигляді таблиці.
5. Зберегти результат виконання програми

**Теоретичні відомості**

В програмуванні часто виникають задачі для вирішення яких часто приходиться застосовувати один і той самий алгоритм. Для економії місця в пам’яті слід записати його лише один раз і використовувати його як так звану підпрограму, або функцію, як її називають в мовах вищого рівня і забезпечити наявність точок входу і виходу з підпрограми.

Для виклику підпрограми в моделі комп’ютера слід вказати її початковий адрес в пам’яті та передати їй необхідні параметри, тобто вихідні дані які будуть оброблюватися. Викликається підпрограма командою CALL в якій і вказується потрібний адрес.

Щоб вийти з підпрограми, слід зберегти точку для повернення. Це робиться автоматично, а повернутися в неї можна командою RET.

Для реалізації повернення в тіло програми адрес повернення слід розміщувати в стеку. Стек — спеціальним чином організована безадресна пам’ять доступ до якої виконується через першу комірку яка називається верхівкою стека. При записі туди слово заноситься у верхівку, при чому всі інші слова зміщуються вниз. При зчитуванні зчитується верхнє слово, при чому воно видаляється зі стека і всі інші слова зміщуються вгору.

**Хід роботи:**

**Варіант №3 (13)**

Розробити програму що знаходить різницю сум парних і непарних елементів масиву.

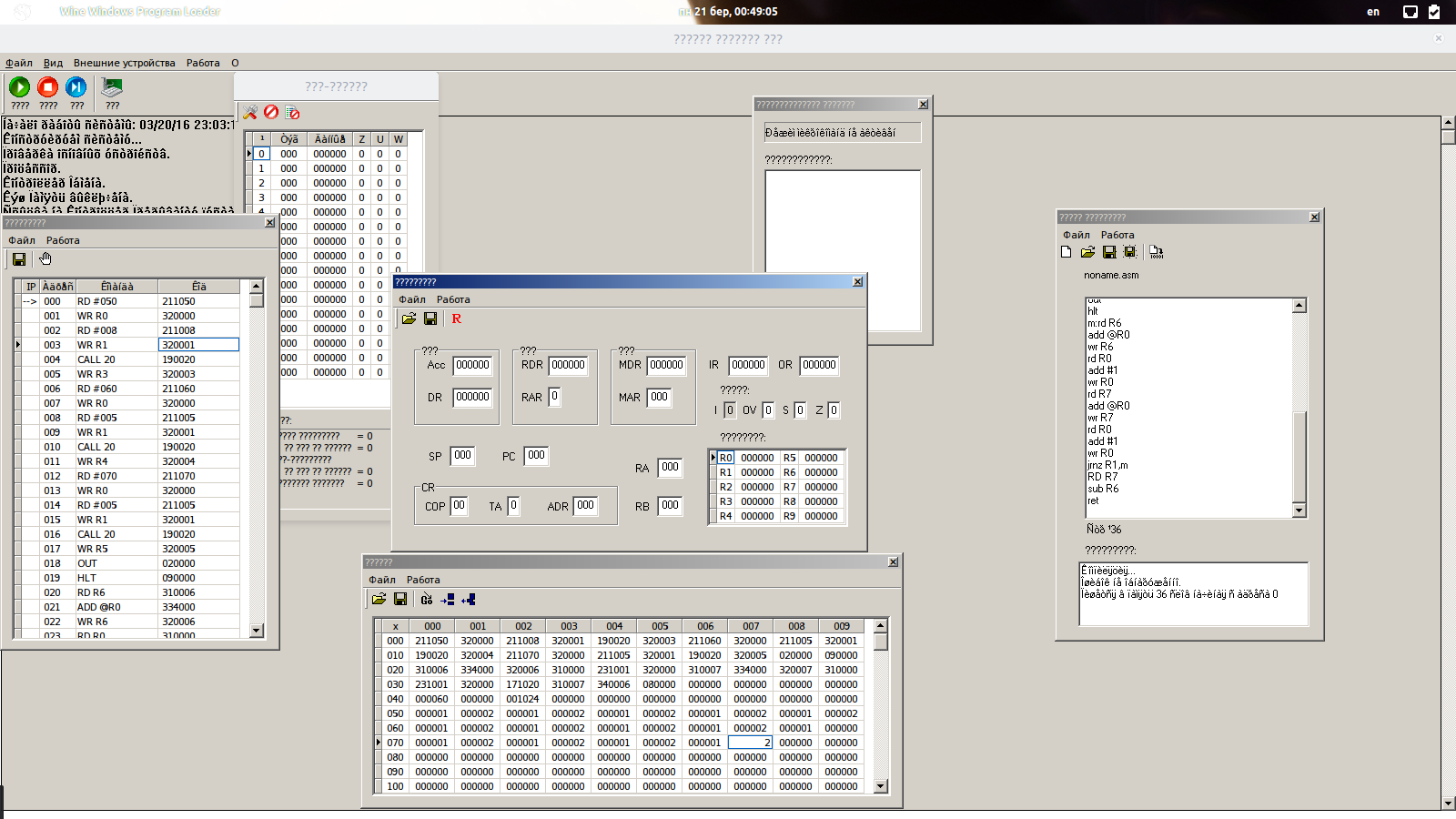
aksanya4.png

Рис.1 Граф-схема алгоритму

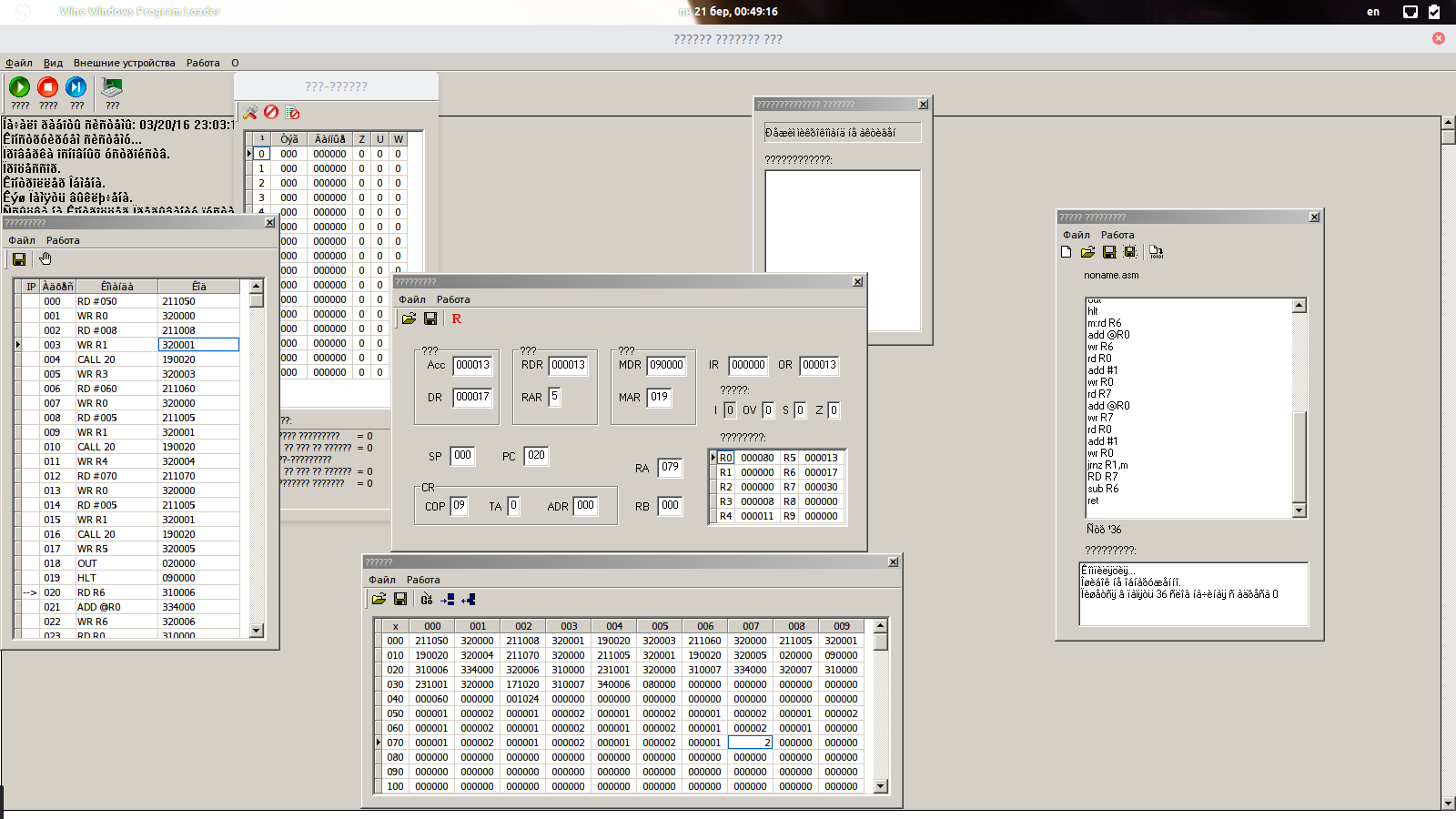
Таблиця 1 – **Текст програми**

|  |  |  |
| --- | --- | --- |
| Адрес | Команда | Коментарі |
| 000 | rd #50 | Зчитуємо адрес першої комірки 1 масиву |
| 001 | wr R0 | Запишемо в R0 |
| 002 | rd #8 | Зчитиуємо довжину 1 масиву |
| 003 | wr R1 | Запишемо в R1 |
| 004 | call m | Викликаємо підпрограму |
| 005 | wr R3 | Записуємо результат в R3 |
| 006 | rd #60 | Зчитуємо адрес першої комірки 2 масиву |
| 007 | wr R0 | Запишемо в R0 |
| 008 | rd #5 | Зчитиуємо довжину 2 масиву |
| 009 | wr R1 | Запишемо в R1 |
| 010 | call m | Викликаємо підпрограму |
| 011 | wr R4 | Записуємо результат в R4 |
| 012 | rd #70 | Зчитуємо адрес першої комірки 3 масиву |
| 013 | wr R0 | Запишемо в R0 |
| 014 | rd #5 | Зчитиуємо довжину 3 масиву |
| 015 | wr R1 | Запишемо в R1 |
| 016 | call m | Викликаємо підпрограму |
| 017 | wr R5 | Записуємо результат в R5 |
| 018 | out | Вихід з програми |
| 019 | hlt | Вхід в підпрограму |
| 020 | m:rd R6 | Зчитування суми всіх непарних елементів |
| 021 | add @R0 | Додавання до неї поточного числа |
| 022 | wr R6 | Перезапис |
| 023 | rd R0 | Зчитування адреси поточного елемента |
| 024 | add #1 | Збільшення її на 1 |
| 025 | wr R0 | Перезапис |
| 026 | rd R7 | Зчитування суми всіх непарних елементів |
| 027 | add @R0 | Додавання до неї поточного числа |
| 028 | wr R7 | Перезапис |
| 029 | rd R0 | Зчитування адреси поточного елемента |
| 030 | add #1 | Збільшення її на 1 |
| 031 | wr R0 | Перезапис |
| 032 | jrnz R1,m | Якщо не досягнуто кінця масиву, повернення до m |
| 033 | RD R6 | Інакше зчитування суми парних |
| 034 | sub R7 | Віднімання від неї суми непарних |
| 035 | ret | Повернення в програму |

**Вид моделі до запуску програми:**



**Результат виконання програми:**



**Висновок**

На лабораторній роботі я створив програму в середовищі моделі комп’ютера. Програма знаходить різницю сум парних та непарних елементів в трьох масивах використовуючи регістри процесора та оператор CALL.