ЛР № 6. Дослідження виконання інструкцій симулятора MARIE за допомогою DataPath

Мета: зрозуміти і дослідити принцип виконання інструкцій симулятора MARIE

Завдання: розробити програму, завантажити програму до симулятора, виконати програму в автоматичному режимі і покроковому режимах, проаналізувати і пояснити отримані результати; дослідити виконання інструкцій використовуючи DataPath; скласти звіт з виконання лабораторних досліджень та захистити його.

Методика виконання лабораторної роботи

Симулятор тракту даних MARIE DataPath показує переміщення даних на рівні регістрів комп'ютера MARIE. На рисунку 1 зображено робоче вікно DataPath, яке складається з наступних частин: рядок меню, графічне зображення регістрів тракту даних, область Machine Monitor і внизу знаходиться область повідомлень.

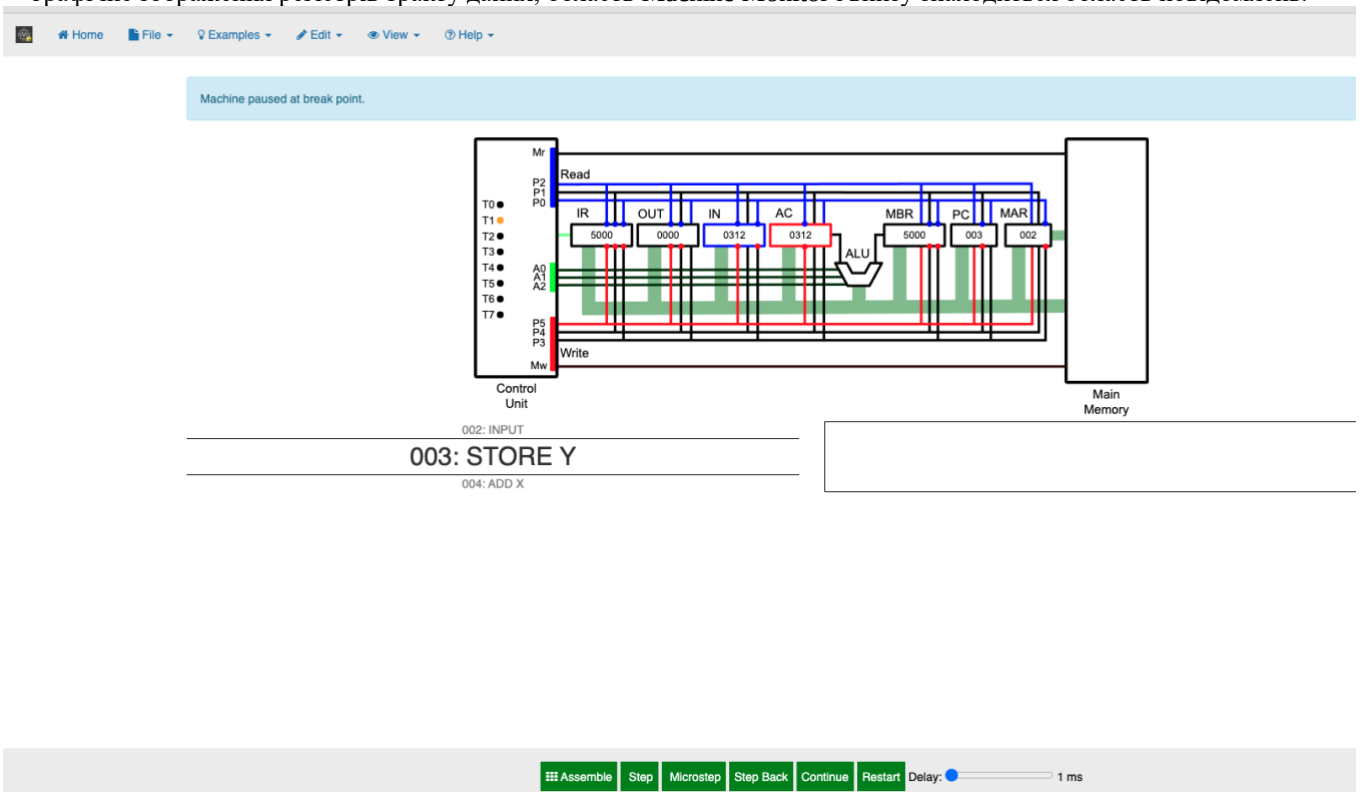

Рис 6.1. Робоче вікно симулятора MarieDPath

DataPath має менше можливостей, ніж MarieSim, оскільки його метою є показати рух даних в межах регістрів комп’ютера Marie. Як і в MarieSim, є можливість завантаження файлів програми, перезапуск програми і перезапуску системи.

В DataPath будуть працювати тільки ті програми, які були відасеммбльовані симулятором Marie. При виконанні програми в графічній частині де зображені регістри підсвічується по черзі той регістр, який задіянийна певному етапі виконання інструкції.

У ході виконання інструкції, в правому нижньому кутку відображаються стани регістрів для кожного такту.

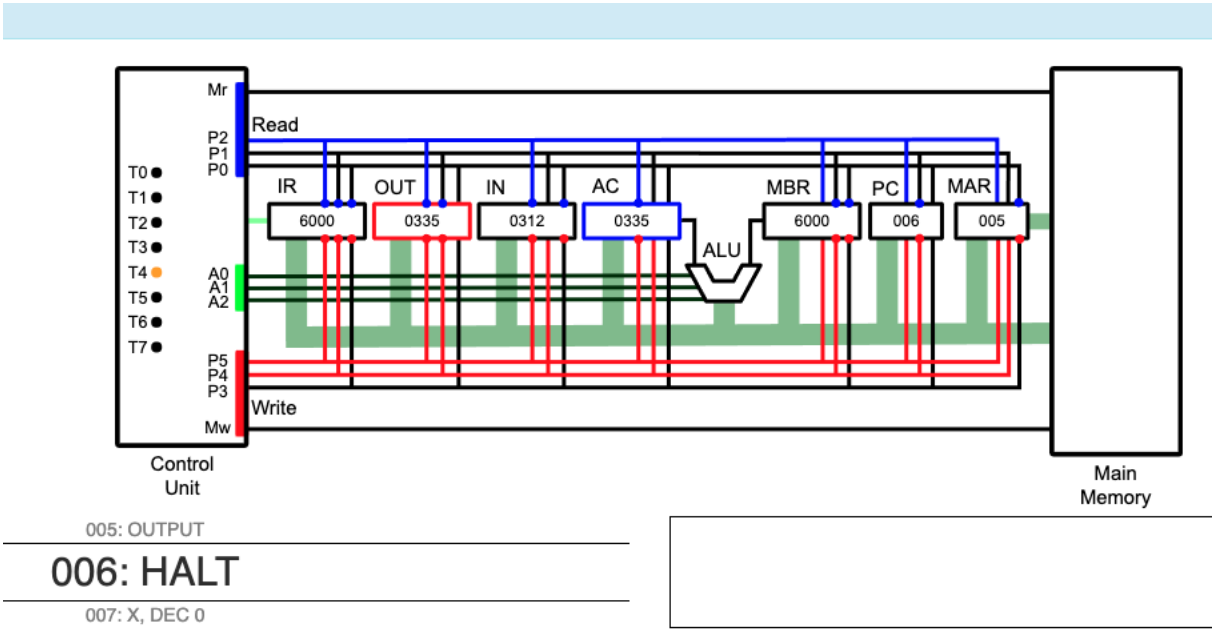

Рис 6.2 Робоче вікно симулятора MarieDPath під час виконання програми

Завдання

Значення x, y, z вводяться з клавіатури, результат записується в пам’ять і виводиться на екран. Програму реалізовувати за домогою циклу (використання інструкцій умовного переходу). Персональні варіанти завдань знайдете у табличці з варіантами.

Додаток до лабораторних робіт 5-6.

Директиви асемблера

В асемблері MARIE використовуються чотири директиви:

ORG – визначає початкову адресу вашої програми. Якщо ви не включите директиву ORG в коді, перша адреса вашої програми автоматично стане 000h.

Інші три директиви дозволяють визначити константи у вашій програмі в десятковій (DEC), вісімковій (OCT) і шістнадцятковій (HEX) формі. Розмір слова машини MARIE становить 16 біт, таким чином, допустимий діапазон десяткових констант -32768 до +32767 (8000h до 7FFFh).

Операнди

Всі операнди в машині MARIE є адресами, допустимимі значення є від 000h до FFFh, оскільки в машині MARIE є 4096 комірок пам'яті. Ці адреси повинні бути приведені у шістнадцятковій формі, з нулем на початку. Ви також можете використовувати мітки для зберігання операндів. Отже, якщо ви хочете додати значення за адресою F00 до акумулятора, можна використовувати інструкцію, ADD 0f00. Якщо замість цього використовувати інструкцію, ADD F00, асемблер буде шукати мітку F00 десь у вашій програмі. Якщо у вас немає такої мітки у вашій програмі, асемблер буде видавати помилку. Ви можете використовувати мітки для зберігання операндів.

Система інструкції (Також доступна за посилання https://github.com/MARIE-js/MARIE.js/wiki/MARIE-Instruction-Set-(with-Opcodes))

Мнемоніка	Hex	Опис
Add X	3	Додавання вмістимого комірки X до AC
AddI X	B	Значення за адресою X використовується в якості актуальної адреси операнда, який додається до AC
Clear	A	Обнулення AC
Input	5	Занесення значення з клавіатури в AC
Halt	7	Зупинка виконання програми
Jump X	9	Занесення значення за адресою X в PC
JumpI X	C	Значення за адресою X використовується в якості актуальної адреси операнда, який додається до PC
JnS X	0	Збереження значення PC в комірку X і перехід PC=X+1
Load X	1	Завантаження значення з комірки X в AC
Output	6	Вивід значення з AC на екран
Skipcond X	8	Пропустити наступну інструкцію, коли виконалася умова
Store X	2	Збереження значення з AC в комірку X
Subt X	4	Відняти від AC значення за адресою X

Зауваження до використання інструкції SKIPCOND:
Два адресні біти найближчі до поля команди, 10 та 11 біти визначають умову на яку буде відбуватися тест. Якщо два адресні біти є 00, це означає – «пропустити, якщо значення AC від'ємне». Якщо адресні біти 01, це означає – «пропустити якщо значення AC = 0». Якщо адресні біти 10, це означає проустити, якщо AC більше 0. Наприклад: інструкція Skipcond 800 пропустить наступну інструкцію у випадку, якщо AC більше 0.

Note regarding use of SKIPCOND:

The two address bits closest to the opcode field, bits 10 and 11 specify the condition to be tested. If the two address bits are 00, this translates to "skip if the AC is negative". If the two address bits are 01, this translates to "skip if the AC is equal to 0". Finally, if the two address bits are 10 (or 2), this translates to "skip if the AC is greater than 0".

Example: the instruction Skipcond 800 will skip the instruction that follows if the AC is greater than 0.