Министерство высшего образования и науки Российской Федерации

Национальный научно-исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №7  
по дисциплине  
**«Основы профессиональной деятельности».**

Вариант №696.

Работу выполнил:

Афанасьев Кирилл Александрович,  
Студент группы P3106.  
Преподаватель:  
Афанасьев Дмитрий Борисович.

Санкт-Петербург, 2023

Оглавление

[Задание 3](#_Toc138286254)

[Текст исходной программы 3](#_Toc138286255)

[Описание программы 5](#_Toc138286256)

[Методики проверки 6](#_Toc138286257)

[Проверка основного цикла 6](#_Toc138286258)

[Проверка прерывания ВУ-2 7](#_Toc138286259)

[Проверка прерывания ВУ-3 7](#_Toc138286260)

[Сводная таблица результатов проверок по соответствующим методикам 8](#_Toc138286261)

[Вывод 8](#_Toc138286262)

Задание

«Синтезировать цикл исполнения для выданных преподавателем команд. Разработать тестовые программы, которые проверяют каждую из синтезированных команд. Загрузить в микропрограммную память БЭВМ циклы исполнения синтезированных команд, загрузить в основную память БЭВМ тестовые программы. Проверить и отладить разработанные тестовые программы и микропрограммы.

Вариант 696:

1. MSUB M - вычитание аккумулятора из ячейки с записью результата в ячейку памяти без установки N/Z/V/C
2. Код операции - 9...
3. Тестовая программа должна начинаться с адреса 040D16

»

Текст синтезированной команды

*Таблица 1: Текст синтезированной команды.*

|  |  |  |
| --- | --- | --- |
| Адрес МП | Микрокоманда | Действие ; Комментарий |
| E0  E1  E2  E3  E4  E5  E6  E7  E8  E9 | 8xE4084002  8xE4044002  8xE4024002  8xEA014002  8xE8  8xE8  8xE8  8xEA  0001009411  8055101040 | IF CR(11) = 0 GOTO E4 ; X(0..7)XX -> на E4  IF CR(10) = 1 GOTO E4 ; X(C..F)XX -> на E4  IF CR(9) = 1 GOTO E4 ; X(A..B)XX -> на E4  IF CR(8) = 1 GOTO EA ; X8XX -> на E4, X9XX -> за борт  ~AC + DR + 1 -> DR ; В DR – операнд M - AC.  GOTO STORE @ 55 ; Сохраним результат в память. |

Текст исходной программы

*Таблица 1: Текст исходной программы.*

|  |
| --- |
| **Описание программы снизу** |
| ORG 0x020  START: DI ; *На время инициализации векторов.*  ; *запретим какие-либо прерывания.*  LD #0xA ; *Инициализируем прерывание ВУ-2.*  OUT 0x5 ; *на вектор 2 (и разрешим его).*  LD #0xB ; *Инициализируем прерывание ВУ-3.*  OUT 0x7 ; *на вектор 3 (и разрешим его).*  CLA ; *Остальные прерывания ВУ будут.*  OUT 0x1 ; *назначены на вектор 0.*  OUT 0x3 ; *(и вообще запрещены на уровне КВУ).*  OUT 0xB ; *….*  OUT 0xE ; *….*  OUT 0x12 ; *….*  OUT 0x16 ; *….*  OUT 0x1A ; *….*  OUT 0x1E ; *….*  JUMP $PROG ; *Векторы назначены. Двигаемся в*  ; *основной цикл…* |
| ORG 0x030  ADDR\_X: WORD 0x031 ; *Указатель на X.*  X: WORD ? ; *Основная переменная программы.*  PROG: EI ; *Разрешаем прерывания.*  LD ADDR\_X ; *Загрузим адрес X.*  PUSH ; *И положим адрес в стек.*  INCLP: CALL $AT\_INC ; *Вызываем атомарную операцию +2.*  JUMP INCLP ; *Это вся суть основного цикла.* |
| ORG 0x040  TEMP\_ADDR: WORD ? ; *Врем. ячейка для адреса аргумента.*  AT\_INC: DI ; *Атомарное увел. яч. памяти на 2.*  LD &1 ; *Загрузим адрес операнда.*  ST TEMP\_ADDR ; *Сохраним во временную ячейку.*  LD (TEMP\_ADDR) ; *Загрузим операнд.*  NOP ; *Точка отладки 1.*  INC ; *Увеличим его на 2. Прерывания*  INC ; *были запрещены до этого.*  CALL $AAV\_CHECK ; *Сверим с ОДЗ полученный результат.*  ST (TEMP\_ADDR) ; *Сохраним его в ячейку памяти.*  NOP ; *Точка отладки 2.*  EI ; *Снова разрешим прерывания.*  RET ; *И вернем логику управления.* |
| ORG 0x050 ; *Обработчик вектора прерываний 2*  INT2: NOP ; *Точка отладки 3.*  PUSH ; *Сохраним состояние AC.*  IN 0x4 ; *Произведем чтение с ВУ-2.*  SXTB ; *Расширим знак пришедшего числа.*  NEG ; *Изменим знак числа.*  ST $X ; *Сохраним в X.*  POP ; *Вернем состояние AC.*  NOP ; *Точка отладки 4.*  IRET ; *Выйдем из текущего прерывания.* |
| ORG 0x060 ; *Обработчик вектора прерываний 3.*  INT3: NOP ; *Точка отладки 5.*  PUSH ; *Сохраним состояние AC.*  LD $X ; *Загрузим X в AC.*  ASL ; *Арифметический сдвиг влево (X\*2).*  ADD $X ; *2\*X + X -> AC (3\*X).*  SUB #5 ; *Из 3\*X вычитаем 5.*  OUT 6 ; *Итоговый результат F(X) на ВУ-3.*  POP ; *Вернем состояние AC.*  NOP ; *Точка отладки 6.*  IRET ; *Выйдем из текущего прерывания.* |
| ORG 0x070  AAV\_CHECK: CMP MAX\_VAL ; *Проверим верхнюю границу ОДЗ.*  BEQ AAV\_RET ; *Они равны? Да – на выход.*  BGE RESET ; *Результат больше? Сбрасываем его.*  CMP MIN\_VAL ; *Проверим нижнюю границу ОДЗ.*  BGE AAV\_RET ; *Результат больше? На выход.*  RESET: LD MIN\_VAL ; *Иначе сбросим до минимального числа.*  AAV\_RET: RET ; *Тот самый выход!*  MIN\_VAL: WORD 0xFFD5 ; *Нижняя граница ОДЗ.*  MAX\_VAL: WORD 0x002A ; *Верхняя граница ОДЗ.* |

*Окончание таблицы.*

Описание программы

* Назначение основной программы: увеличение значения ячейки памяти на 2. Нажатие на кнопку «Готов» на ВУ-2 обновит значение ячейки памяти инвертированным значением с РДВУ-2. Нажатие на кнопку «Готов» на ВУ-3 выведет в РДВУ-3 значение F(X) = 3X - 5.
* Описание исходных данных:
  + X – основная ячейка памяти.
  + ОПИ:
    - X – знаковое, 8-разрядное число.
  + ОДЗ:
* Расположение в памяти ЭВМ программы и исходных данных:
  + Программный комплекс располагается в следующих ячейках памяти:
    - Векторы прерываний: между ячейками 000 и 00F включительно
    - Стандартный обработчик прерываний: между ячейками 010 и 01F включительно
    - Инициализация векторов прерываний: между ячейками 020 и 02E включительно
    - Основной цикл программы: между ячейками 030 и 036 включительно
    - Подпрограмма атомарного увеличения ячейки на 2: между ячейками 040 и 04С включительно
    - Обработчик прерываний вектора 2: между ячейками 050 и 05A включительно
    - Обработчик прерываний вектора 3: между ячейками 060 и 06B включительно
    - Подпрограмма проверки на вхождение ячейки в ОДЗ: между ячейками 070 и 078 включительно
  + Исходные данные должны располагаться в ячейках памяти:
    - X – 031
  + Результат работы программы должен располагаться в ячейке памяти 031.
  + В программе используются следующие неизменяемые значения (константы):
    - По адресу 000 – значение 0x0010
    - По адресу 001 – значение 0x0180
    - По адресу 002 – значение 0x0010
    - По адресу 003 – значение 0x0180
    - По адресу 004 – значение 0x0050
    - По адресу 005 – значение 0x0180
    - По адресу 006 – значение 0x0060
    - По адресу 007 – значение 0x0180
    - По адресу 008 – значение 0x0010
    - По адресу 009 – значение 0x0180
    - По адресу 00A – значение 0x0010
    - По адресу 00B – значение 0x0180
    - По адресу 00C – значение 0x0010
    - По адресу 00D – значение 0x0180
    - По адресу 00E – значение 0x0010
    - По адресу 00F – значение 0x0180
    - По адресу 077 – значение 0xFFD5
    - По адресу 078 – значение 0x002A
  + В программе также используются вспомогательные ячейки, находящиеся по адресам 030 и 040.
* Первая команда располагается в ячейке по адресу 020.

Вывод

Во время выполнения данной лабораторной работы я изучил процесс прерывания программы и исследовал функционирование Базовой ЭВМ при обмене данными в режиме прерывания программы, а также научился писать методики проверки программных комплексов и следовать им.