Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет имени

В.Ф. Уткина»

Кафедра «Электронные вычислительные машины»

Отчёт по лабораторной работе №4

**ЭМУЛЯЦИЯ КОМАНД МИКРОПРОЦЕССОРА i8086**

по дисциплине

“ЭВМ и периферийные устройства”

|  |
| --- |

**Выполнили:**

ст. гр. 245

Бригада №2

Бекренев Владислав

Луковкин Иван

**Проверил:**

ст. пр. Устюков Д.И.

ст. пр. Тарасов А.С.

Рязань 2023

**Цель работы:** ознакомление с принципами микропрограммной эмуляции ЭВМ с программным управлением, микропрограммирование машинных команд микропроцессора i8086 (МП-86).

**Теоретическая часть**

АРХИТЕКТУРА ПРОЦЕССОРА И ОСНОВНЫЕ ЭТАПЫ ВЫПОЛНЕНИЯ МАШИННЫХ КОМАНД

В состав процессора входят (рис. 1) регистр команд RGK, арифметико-логическое устройство (АЛУ), устройство управления (УУ) и регистры, доступные программисту на уровне команд программы. Эти регистры можно разделить на четыре группы:

а) регистры общего назначения (РОНы). РОНы разделены на две половины – старшую и младшую. В некоторых командах РОНы специализированны. Это отражается в их названиях:

аккумулятор (AX) – главный рабочий регистр, команды,

относящиеся к нему, являются самыми короткими;

база (BX) – кроме общего назначения, этот регистр используется

также для адресации операндов в памяти;

счетчик (CX) – в некоторых командах содержимое CL или CX

используется как счетчик.

Рис. 1. Архитектура микропроцессора

3

б) указательные (SP – указатель стека, BP – указатель базы) и индексные (SI – индекс источника, DI – индекс получателя) регистры – это 16-битные регистры, которые используются для хранения адресов, а также участвуют в командах как РОНы;

в) сегментные регистры (CS – кода, DS – данных, SS – стека, ES – дополнительных данных). Их необходимость вызвана способом организации памяти. В общем пространстве памяти (1 МБ) в любом месте произвольно находятся 4 сегмента памяти емкостью 64 Кб. Эти сегменты называются:

сегмент кода (программы), содержащий команды программы;

сегмент данных, содержащий данные, обрабатываемые

программой;

сегмент стека, содержащий стек;

сегмент дополнительных данных (экстра-сегмент).

Сегментные регистры содержат начальные адреса этих 4-х сегментов, точнее, 16 старших бит адреса сегмента, а четыре младших бита считаются нулевыми;

г) регистр состояния PSW. Он состоит из программного счетчика PC (указатель IP) и регистра флажков (FLAG). IP содержит адрес следующей команды, выполняемой МП-86. Регистр флажков содержит 9 флажков, 3 из которых являются управляющими и 6 - арифметические, которые отражают признаки результата. Рассмотрим некоторые их флажков:

SF – фиксирует знаковый бит результата;

ZF – фиксирует нулевой результат;

PF – фиксирует четное число единиц результата (флажок

паритета);

CF – фиксирует перенос (заем) из старшего бита результата.

АЛУ необходимо для выполнения указанных в КОП команд арифметических и логических операций над данными.

УУ осуществляет в соответствии с КОП команды выработку внутренних и внешних управляющих сигналов, необходимых для выполнения текущей команды.

RGK содержит код текущей команды на время ее дешифрации и выполнения.

В прототипе МП-86 полный (физический) адрес содержит 20 бит и равен сумме сегментного регистра, сдвинутого на 4 бита влево (сегментный адрес), и 16-битного эффективного адреса (смещение). В модели считаем, что данные и программа находятся в одном сегменте (CS=0000H, DS=0000H).

В общем виде порядок выполнения команд (программы на машинном языке) в процессоре представлен на рис. 2. В регистре IP записан начальный адрес программы в ОП. Микропроцессор обращается к ОП и осуществляет выборку команды по адресу из IP. Затем загружает ее в регистр команд RGK и дешифрирует ее с одновременным инкрементом IP для адресации следующей по порядку команды. Если команда не является командой перехода, то выполняем ее и переходим к следующей по порядку команде. Команды выполняются последовательно до тех пор, пока не встретится команда перехода. Если встречается команда безусловного перехода, то изменяется естественный порядок следования команд путем замещения содержимого IP адресом, определяемым самой командой перехода. Команды условных переходов замещают или не замещают содержимое IP в зависимости от результатов предыдущих команд, т.е. от состояния регистра FLAG. Если, например, после команды «вычитания» находится команда «перехода по нулю», переход осуществляется, если в регистре FLAG флажок ZF=1. Если же ZF=0, то переход не производится. Когда реализован переход, начинается новая последовательность команд с адреса, к которому осуществлен переход.

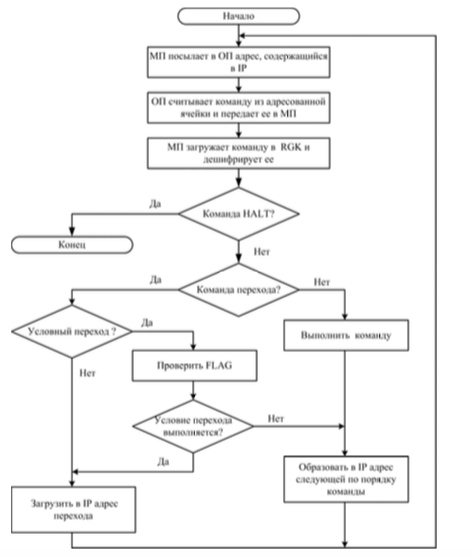


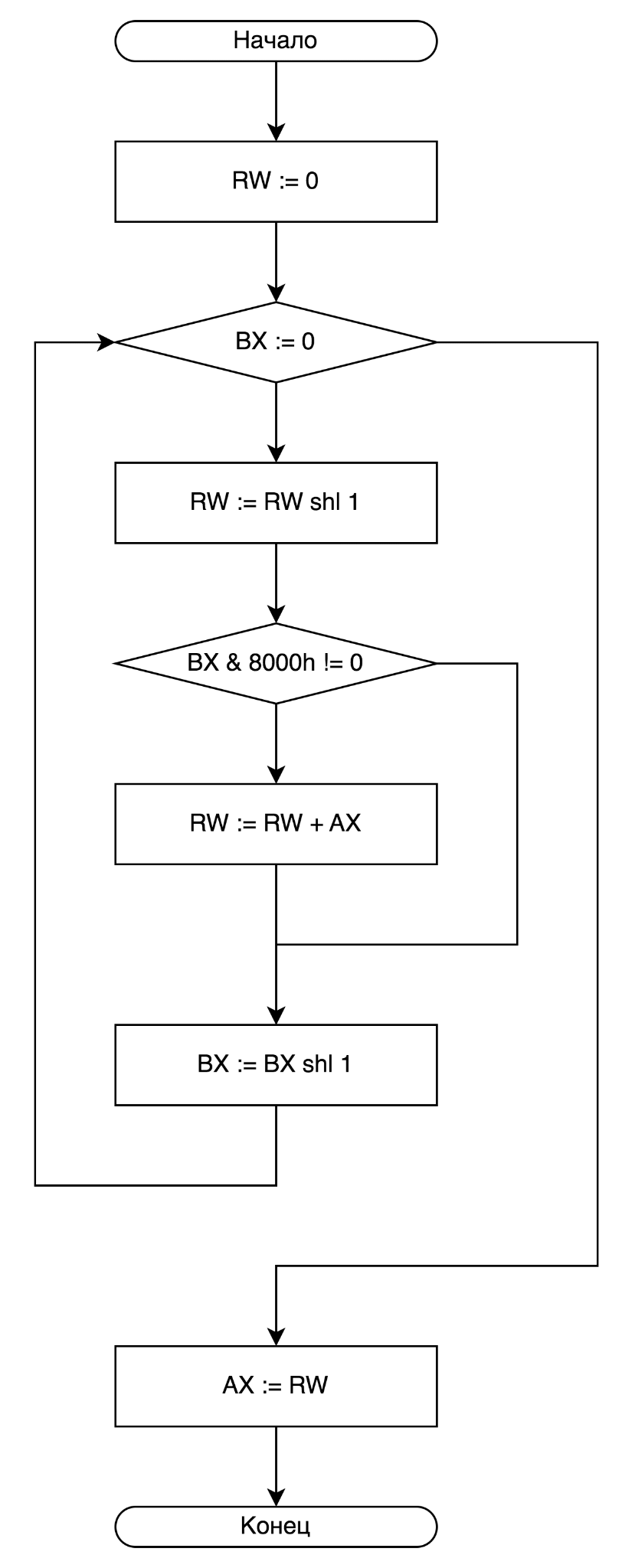
Рис. 2. Последовательное выполнение команд

**Практическая часть**

Задание:

Эмулировать представленные операции:

1. MOV (Непосредственная адресация)
2. PUSH - включить в стек
3. POP - извлечь из стека
4. INC - инкремент (регистровая адресация)
5. ROL - циклический сдвиг влево (на 1 разряд)
6. JMP - безусловный переход
7. MUL - умножение чисел без знака (операнды - байты, результат - слово; умножение с анализом старшего разряда множителя со сдвигом СЧП). Схема алгоритма представлена на рисунке 1а.
8. XLAT - команда преобразования. Схема алгоритма представлена на рисунке 1б.



а) б)

Рисунок 1 - Схемы алгоритмов

а) Реализация операции MUL

б) Реализация операции XLAT

Таблица 1 - Преобразование команд

| **Команда** | **Тип** | **Код команды в двоичном виде** | | | | **Код команды в 16-ричном виде** |
| --- | --- | --- | --- | --- | --- | --- |
| Байт 1  7654 3210 | Байт 2 | | |
| 76 | 543 | 210 |
| **MOV CX,11h\*** | RI | 1011 1reg | - | 001 | - | B900h |
| **PUSH CX** | RM | 1111 1111 | 11 | 110 | 001 | FFF1 |
| **POP BX** | RM | 1000 1111 | 11 | 000 | 011 | 8FC3 |
| **INC CX** | RM | 1111 1110 | 11 | 000 | 001 | FEC1 |
| **ROL 1 BX** | RM | 1101 0001 | 11 | 000 | 011 | D1C3 |
| **JMP** | - | 1110 1011 | - | - | - | EB00 |
| **MUL BX** | RM | 1111 0110 | 11 | 100 | 011 | F6E3 |
| **XLAT** | - | 1101 0111 | - | - | - | D700 |
| **HALT** | - | 1111 1111 | - | - | - | FF00 |

\* - В команде MOV используется константа 11h, добавленная в ячейку dataL.

Таблица 2 - Микропрограмма выборки команд

| **Адрес МК** | **Операция** | **Поле** | **Значение** | **Функция** |
| --- | --- | --- | --- | --- |
| 00 | ARAM := IP  IP := IP + 022 | B  WM  SRC  ALU  DST  CONST | C  3  5  3  4  000202 | IP  ARAM := RGB  CONST, RGB  R + S + C0  Запись в РЗУ |
| 01 | Чтение ОП  RGK := RGR  Дешифрация | MEM  B  DST  CHA | 5  E  1  2 | Чтение слова  RGK  РЗУ[B] := RGR  JMAP |

Таблица 3 - Микропрограммы операций

| **Адрес МК** | **Операция** | **Поле** | **Значение** | **Функция** |
| --- | --- | --- | --- | --- |
| 03 | **MOV CX, 1111h**  IP := IP + 2 | B  SRC  ALU  DST  DST  WM  CONST | C  5  3  4  4  2  0002 | IP  CONST, RGB  R + S + C0  РЗУ[В] := SDA  РЗУ[B] := SDA  SDA := ARAM |
| 04 | Чтение ОП  CX := RGR | B  MEM  DST  CHA | 1  5  1  0 | CX  Чтение слова  РЗУ[B] := RGR  JZ |
| 05 | **PUSH CX** | A  B  DST  CHA | 0  4  4  0 | CX  SP  РЗУ[B] := SDA  JZ |
| 06 | **POP BX** | A  B  DST  CHA | 4  3  4  0 | SP  BX  РЗУ[B] := SDA  JZ |
| 07 | **INC CX** | B  ALU  CCX  DST  CHA | 1  4  1  4  0 | CX  S + C0  РЗУ[B] := SDA  JZ |
| 08 | **ROL 1** | B  SH  N  ALU  DST  JFI  CC  CHA  CONST | 1  8  1  4  4  1  3  3  000A | CX  shl 1  S + C0  РЗУ[B] := SDA  I = 1  JO  Переход по адресу  Адрес перехода |
| 09 | reg := reg + 1 | B  ALU  CCX  DST | 1  4  1  4 | CX  S + C0  РЗУ[B] := SDA |
| 0A |  | CHA | 0 | JZ |
| 0B | **JMP** | JFI  CHA  CONST | 4  3  0007 | Б/у переход  Переход по адресу  Адрес перехода |
| 0C | **HALT** | JFI | 5 | STOP |
| 0D | **XLAT**  RW := AX | A  B  DST | 0  F  4 | AX  RW  РЗУ[B] := SDA |
| 0E | RW & 00FFh  ARAM := SDA | B  SRC  ALU  WM  CONST | F  5  9  2  00FF | RW  CONST, RGB  R & S  SDA := ARAM |
| 0F | RW := RGR | B  MEM  DST | F  4  1 | RW  Чтение байта  РЗУ[B] := RGR |
| 10 | AX & FF00h | B  SRC  ALU  DST  CONST | 0  5  9  4  FF00 | AX  CONST, RGB  R & S  РЗУ[B] := SDA |
| 11 | AX := AX + RW | A  B  ALU  DST  CHA | F  0  3  4  0 | RW  AX  R + S + C0  РЗУ[B] := SDA  JZ |
| 12 | **MUL**  RW := 0 | B  SRC  DST  CONST | F  5  4  0 | RW  CONST, RGB  РЗУ[B] := SDA |
| 13 | BX = 0  if Z then goto END | A  CC  CHA  CONST | 3  1  3  19 | BX  JZ  Переход по команде  Адрес перехода |
| 14 | RW := RW shl 1 | B  SH  N  ALU  DST | F  8  1  4  4 | RW  shl  R + C0  РЗУ[B] := SDA |
| 15 | BX & 8000h | B  SRC  ALU  F  CONST | 3  5  9  1  8000 | BX  CONST, RGB  R & S  Сохранение флагов в RFD  Маска |
| 16 | if Z(RFD) == 1  goto M2 | ALU  JFI  CC  CHA  CONST | 0  2  1  3  18 | На выходе все нули  Проверка по флагам RFD  JZ  Переход по адресу  Адрес перехода |
| 17 | RW := RW + AX | A  B  ALU  DST | 0  F  3  4 | AX  RW  R + S + C0  РЗУ[B] := SDA |
| 18 | M2:  BX shl 1 | B  SH  N  DST  ALU  JFI  CHA  CONST | 3  8  1  4  4  4  3  0013 | BX  shl  РЗУ[B] := SDA  S + C0  Б/у переход  Переход по адресу  Адрес перехода |
| 19 | AX := RW | A  B  DST  CHA | F  0  4  0 | RW  AX  РЗУ[B] := SDA  JZ |

Таблица 4 - Таблица преобразования адресов в программе

| **Начальный адрес:** | **Код операции:** |
| --- | --- |
| 03 | 10111001XXXXXXXX |
| 05 | 1111111111110001 |
| 06 | 1000111111000011 |
| 07 | 1111111011000001 |
| 08 | 1101000111000011 |
| 0B | 11101011XXXXXXXX |
| 0C | 11111111XXXXXXXX |
| 0D | 11010111XXXXXXXX |
| 12 | 1111011011100011 |

Содержимое ОЗУ представлено на рисунке 2.

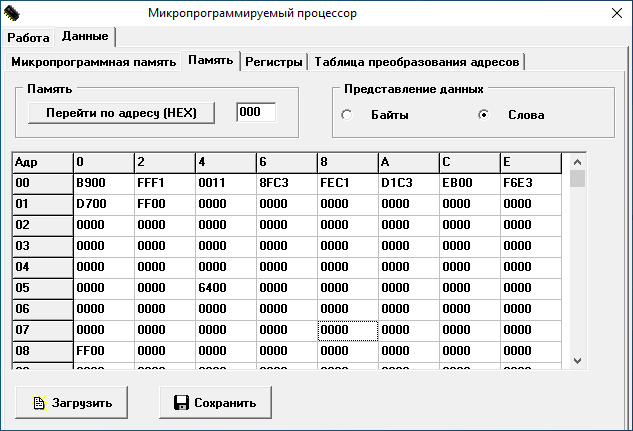


Рисунок 2 - Содержимое ОЗУ

Исходные данные представлены на рисунке 3.

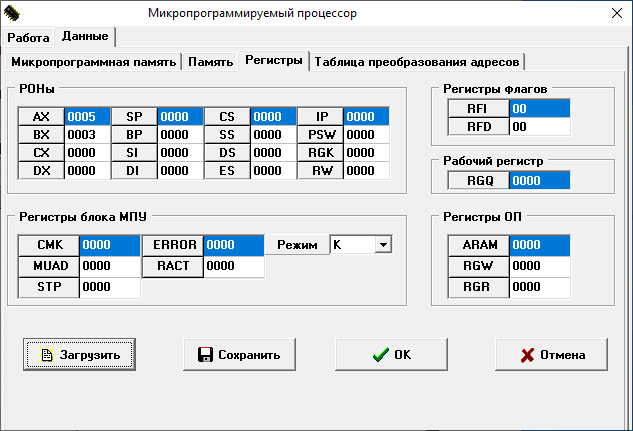


Рисунок 3 - Исходные данные

Процесс изменения регистров (трасса) при выполнении команд:

Таблица 1 - трассировка команд

| **MK/K** | **РОНы** | | | **Спец. регистры** | | **RGA** | **RGB** | **RGR** | **ALU** | **SDA** | **RGK** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **AX** | **CX** | **BX** | **RW** | **IP** |
| pre | 0005 | 0000 | 0003 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| K1 | 0005 | **0011** | 0003 | 0000 | **0004** | 0000 | 0000 | 0000 | **0005** | **0005** | **B900** |
| K2 | 0005 | 0011 | 0003 | 0000 | **0006** | 0000 | 0000 | **FFF1** | 0005 | 0005 | **FFF1** |
| K3 | 0005 | 0011 | **0005** | 0000 | **0008** | 0000 | **0003** | **8CF3** | 0005 | 0005 | **8CF3** |
| K4 | 0005 | **1112** | 0005 | 0000 | **000A** | 0000 | **0011** | **FEC1** | **1112** | **1112** | **FEC1** |
| K5 | 0005 | **2224** | 0005 | 0000 | **000C** | 0000 | **0005** | **D1C3** | **2224** | **2224** | **D1C3** |
| K6 | 0005 | **2225** | 0005 | 0000 | **000E** | 0000 | **2224** | **EB00** | **2225** | **2225** | **EB00** |
| K7 | **0019** | 2225 | **0000** | **0019** | **0010** | **0019** | **0005** | **F6E3** | **0019** | **0019** | **F6E3** |
| K8 | **0064** | 2225 | 0000 | **0064** | **0012** | **0064** | 0005 | **0064** | **0064** | **0064** | **D700** |

Трассировка команды MUL:

| **MK/K** | **РОНы** | | | **Спец. регистры** | | **RGA** | **RGB** | **RGR** | **ALU** | **SDA** | **RGK** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **AX** | **CX** | **BX** | **RW** | **IP** |
| pre | 0005 | 2225 | 0005 | 0000 | 000E | 0005 | 2224 | **EB00** | 2225 | 2225 | **EB00** |
| K7 | 0005 | 2225 | 0005 | 0000 | **0010** | 0005 | 2224 | **F6E3** | 2225 | 2225 | **F6E3** |
| 12 | 0005 | 2225 | 0005 | **0000** | 0010 | 0005 | **0000** | F6E3 | **0000** | **0000** | F6E3 |
| 13 | 0005 | 2225 | 0005 | 0000 | 0010 | 0005 | **0005** | F6E3 | **0005** | **0005** | F6E3 |
| 14 | 0005 | 2225 | 0005 | 0000 | 0010 | 0005 | **0000** | F6E3 | **0000** | **0000** | F6E3 |
| 15 | 0005 | 2225 | 0005 | 0000 | 0010 | 0005 | **0005** | F6E3 | 0000 | 0000 | F6E3 |
| 16 | 0005 | 2225 | 0005 | 0000 | 0010 | 0005 | 0005 | F6E3 | 0000 | 0000 | F6E3 |
| 18 | 0005 | 2225 | **000A** | 0000 | 0010 | 0005 | 0005 | F6E3 | **0005** | **000A** | F6E3 |
| 13 | 0005 | 2225 | 000A | 0000 | 0010 | **000A** | 0005 | F6E3 | **000A** | 0000 | F6E3 |
| 14 | 0005 | 2225 | 000A | 0000 | 0010 | **0005** | **0000** | F6E3 | **0000** | **0000** | F6E3 |
| 15 | 0005 | 2225 | 000A | 0000 | 0010 | 0005 | **000A** | F6E3 | 0000 | 0000 | F6E3 |
| 16 | 0005 | 2225 | 000A | 0000 | 0010 | 0005 | **0005** | F6E3 | 0000 | 0000 | F6E3 |
| 18 | 0005 | 2225 | **0014** | 0000 | 0010 | **000A** | 0005 | F6E3 | **000A** | 0000 | F6E3 |
| 13 | 0005 | 2225 | 0014 | 0000 | 0010 | **0014** | **0005** | F6E3 | **0014** | 0000 | F6E3 |
| 14 | 0005 | 2225 | 0014 | 0000 | 0010 | **0005** | **0000** | F6E3 | **0000** | **0000** | F6E3 |
| 15 | 0005 | 2225 | 0014 | 0000 | 0010 | 0005 | **0014** | F6E3 | 0000 | 0000 | F6E3 |
| 16 | 0005 | 2225 | 0014 | 0000 | 0010 | 0005 | **0005** | F6E3 | 0000 | 0000 | F6E3 |
| 18 | 0005 | 2225 | **0028** | 0000 | 0010 | 0005 | **0014** | F6E3 | **0014** | **0028** | F6E3 |
| … | 0005 | 2225 | 0028 | 0000 | 0010 | 0005 | 0014 | F6E3 | 0014 | 0028 | F6E3 |
| 18 | 0005 | 2225 | **5000** | 0000 | 0010 | 0005 | **5000** | F6E3 | **5000** | **A000** | F6E3 |
| 13 | 0005 | 2225 | 5000 | 0000 | 0010 | **A000** | **0005** | F6E3 | **A000** | A000 | F6E3 |
| 14 | 0005 | 2225 | 5000 | 0000 | 0010 | **0005** | **0000** | F6E3 | **0000** | **0000** | F6E3 |
| 15 | 0005 | 2225 | 5000 | 0000 | 0010 | 0005 | **A000** | F6E3 | **8000** | **8000** | F6E3 |
| 16 | 0005 | 2225 | 5000 | 0000 | 0010 | 0005 | **0005** | F6E3 | **0000** | **0000** | F6E3 |
| 17 | 0005 | 2225 | 5000 | **0005** | 0010 | 0005 | **0000** | F6E3 | **0005** | **0005** | F6E3 |
| 18 | 0005 | 2225 | 5000 | 0005 | 0010 | 0005 | **A000** | F6E3 | **A000** | **4000** | F6E3 |
| … | 0005 | 2225 | 5000 | 0005 | 0010 | 0005 | A000 | F6E3 | A000 | 4000 | F6E3 |
|  | **0019** | 2225 | 5000 | 0005 | 0010 | **0019** | **0005** | F6E3 | **0019** | **0019** | F6E3 |

**Заключение**

В ходе выполнения лабораторной работы нам удалось эмулировать команды ассемблера на микрокомандном уровне.