# Инструкция по решению лабы 6 по ЭВМ

Список ссылок по теме:

* Книга «**Основы теории и организации ЭВМ**» В.В. Гуров, В.О. Чуканов.

В бумажном виде из библиотеки МИФИ лекции 1.12 (стр. 110) и первая половина лекции 2.6 (стр. 174), лекции 2.7 (стр. 184), 2.10 (стр. 208)

* В электронном виде тут <https://drive.google.com/open?id=1NB5mPfH1T_lhrGLB1KU4ZbMlS9BDXEmr> книга «**Архитектура и организация ЭВМ**» первая половина лекции на странице 56, лекции на страницах 72, 104 — там почти то же самое, что и в бумажном варианте
* На ИНТУИТе курсы В. В. Гурова. Вот ссылки по теме
  + <https://www.intuit.ru/studies/courses/60/60/lecture/1776>
  + <https://www.intuit.ru/studies/courses/60/60/lecture/1778>
  + <https://www.intuit.ru/studies/courses/60/60/lecture/1784>
* Справка в лабе 6

Я прочитал всё это и пару часов потратил на изучение лабы 6. Далее изложу то, что понял в результате. Если что, за эту лабу у меня 5

В процессоре intel 8086 различаются три типа команды:

* Команды между двумя регистрами или регистром и ячейкой памяти — R/M R
* Команды между регистром и непосредственным операндом, или между ячейкой памяти и непосредственным операндом — R/M I
* Команды между регистром AX и непосредственным операндом — A I

Блок формирования адреса операнда ведёт себя по-разному на этих типах команд

Команда выполняется в 5 этапов:

1. Передача команды из ОЗУ в регистр команд (РК)
2. Передача операнда в АЛУ в первый буферный регистр (БР1)
3. Передача операнда в АЛУ в второй буферный регистр (БР2)
4. Передача кода операции (КОП) из регистра команд в АЛУ

4,5. Выполнение операции в АЛУ

1. Передача результата операции (РЕЗ) в нужное место и заполнение регистров флагов (FLAGS). Вообще флаги называются признаками результата (ПР) и передаются в регистр признаков (а не флагов)

Теперь подробнее про эти этапы

## 1)Передача команды из ОЗУ в РК

1. Отправить эффективный адрес команды, который хранится в ячейке IP, в сумматор физического адреса команды (ΣФАК)
2. Отправить значение сегментного регистра команд (CS) в ΣФАК. Не забудьте умножить это значение на 16
3. Сложить два значения внутри ΣФАК
4. Отправить полученное 20-разрядное значение в регистр адреса (РА)
5. Из РА в дешифратор адреса (ДШ)
6. Из ДШ в ОЗУ
7. Из ОЗУ в соответствующую ячейку РК
8. Передать некоторую информацию в коммутатор операции (КОП). В лабе нужно отмечать все стрелочки, ведущие в КОП
9. Из КОП передать некоторую информацию в блок управления операциями (БУОП) и в IP отправить сигнал на получение следующего байта команды (в IP значение увеличивается на 1). Повторить действия 1-9

## 2)Передача операнда в АЛУ в БР1

Здесь на передний план выходит блок формирования адреса операнда (БФАО)

Для типов R/M R и R/M I алгоритм одинаковый:

1. Передать в БФАО поля **w, md** и **r/m**
2. Передать из тактового генератора (ТГ) сигнал в БФАО, чтобы заставить его работать
3. Передать из БФАО сигналы куда следует:

* Если у нас обращение к регистру, то только передаём сигнал в регистровую память (РП) к нужному регистру, а оттуда на БР1
* Если нужно обратиться к ОЗУ, то передаём сигналы в регистровую память (РП) во все нужные регистры, включая сегментный (!). Ещё нужно передать сигнал в ОЗУ, чтобы подготовить его к работе. Дальше значения из регистров передаются в сумматор физического адреса данных (ΣФАД), не забудьте сегментный регистр умножить на 16. Далее физический адрес передаётся в РА, потом в ДШ и в ОЗУ. ОЗУ передаёт операнд в БР1
* Если у нас есть смещение (Disp), то БФАО передаёт сигнал в Схему считывания… (ССКСиНО), которая получает код смещения и передаёт его в ΣФАД

Для типа A I алгоритм попроще (при A I поля md и r/m отсутствуют):

1. Передать из ТГ сигнал в БФАО
2. Передать из БФАО сигнал в РП к регистру AX
3. Передать значение регистра AX в БР1

## 3)Передача операнда в АЛУ в БР1

Теперь у нас будет три алгоритма

Для типа R/M R

1. Передать в БФАО поля **w** и **reg**
2. Передать из ТГ сигнал в БФАО
3. Передать из БФАО сигнал в РП к нужному регистру
4. Передать из нужного регистра значение в БР2

Для типов R/M I и A I алгоритмы схожи

1. Для R/M I передать в БФАО поля **s** и **w**. Для A I передать в БФАО поле **w**
2. Передать из ТГ сигнал в БФАО
3. Передать из БФАО сигнал на схему (ССКСиНО)
4. Передать непосредственный операнд (Im)в схему
5. Передать полученное значение из схемы в БР2

## 4)Передача кода операции (КОП) из РК в АЛУ

1. Отмечаем все стрелки, которые ведут из РП в коммутатор операции (КОП)
2. Передаём это в блок управления операциями (БУОП)
3. Вырабатываем импульс в датчике сигналов (ДС) и передаём его в БУОП
4. Из БУОП информация о команде отправляется в АЛУ

## 5)Передача РЕЗ в нужное место и ПР в FLAGS

Для типа R/M R в БФАО нужно передать поле **d** в БФАО

Если записать результат нужно в регистр, то алгоритм такой:

1. Передать из ТГ сигнал в БФАО
2. Передать из БФАО сигнал в РП к нужному регистру
3. Передать РЕЗ в РП к нужному регистру
4. Передать ПР в FLAGS

Если записать результат нужно в ячейку ОЗУ, то алгоритм такой:

1. Передать из ТГ сигнал в БФАО
2. Передать из БФАО сигнал в ОЗУ
3. Передать из РА сигнал в ДШ, а оттуда в ОЗУ
4. Передать РЕЗ в ОЗУ
5. Передать ПР в FLAGS

## Что же значат эти загадочные названия полей (ИМХО)

**w (word)** — определяет, работаем мы со словами или с байтами. Кстати, если в команде возникает надпись **byte ptr**, то мы работаем с байтами, а если **word ptr**, то работаем со словами. При выполнении команд типа A I указывает на разрядность непосредственного операнда

**reg (register)** — используется при выполнении команд типа R/M R и указывает на нужный регистр для БР2. Имеет 8 значений, а регистров у нас 16 (считая байтовые). Посему, работает в паре с **w**, тогда возможно закодировать все 16 регистров

**r/m (register/memory)** — определяет нужный регистр для БР2 или режим адресации к ОЗУ. Используется всегда в паре с **md (modifier)**

**d (direction)** — используется при выполнении команд типа R/M R и указывает, в какой из двух адресов положить результат операции

**s** — используется при выполнении команд типа R/M I и указывает разрядность непосредственного операнда. Тут одного **w** недостаточно, потому что существуют команды, в которым происходит работа со словами, а непосредственный операнд имеет разрядность в 1 байт. Поэтому и требуется ещё один бит **s**