Национальный Технический Университет Украины
«Киевский Политехнический Институт»

Домашняя модульная контрольная работа №2
по архитектуре компьютеров

Выполнил
Студент группы ИВ-81
Кушниренко Александр

№ зачетной книжки 8127

Киев 2010

Содержание

1. Техническое задание

Разработать программу вычисления функции: **A = X * (Y & Z)**, операнды вводятся из устройств ввода. Результат (32 разряда) выводится в устройство вывода (последовательность вводимых и выводимых данных определяется разработчиком).

Все исходные данные (форматы команд, форма представления операндов, адреса устройств и т.д.) определяются вариантами заданий. Недостающие данные выбираются самостоятельно.

Все команды одноадресные с прямой адресацией операндов.

Способ умножения 4. Начальные адреса представлены ниже.

2. Описание архитектуры и схема алгоритма вычислений

Таблица 2.1 Модель программиста

R0-R6	Не используются
R7	Счетчик команд
R8	Регистр команд
R9	Регистр маски
R10	Счетчик тактов для умножения
R11 - R12	Рабочие регистры
R13	Верхние биты результата
R14	Нижние биты результата
R15	Аккумулятор

Таблица 2.2 Карта распределения памяти

...	
0002h - 0024h	Программа
...	
0031h - 0037h	Программные данные

Таблица 2.3 Адресное пространство внешних устройств

4h	РС ВУ №3 (in)
6h	РД ВУ №3
...	
12h	РС ВУ №4 (out)
14h	РД ВУ №4
16h	РС ВУ №6 (out)
18h	РД ВУ №6

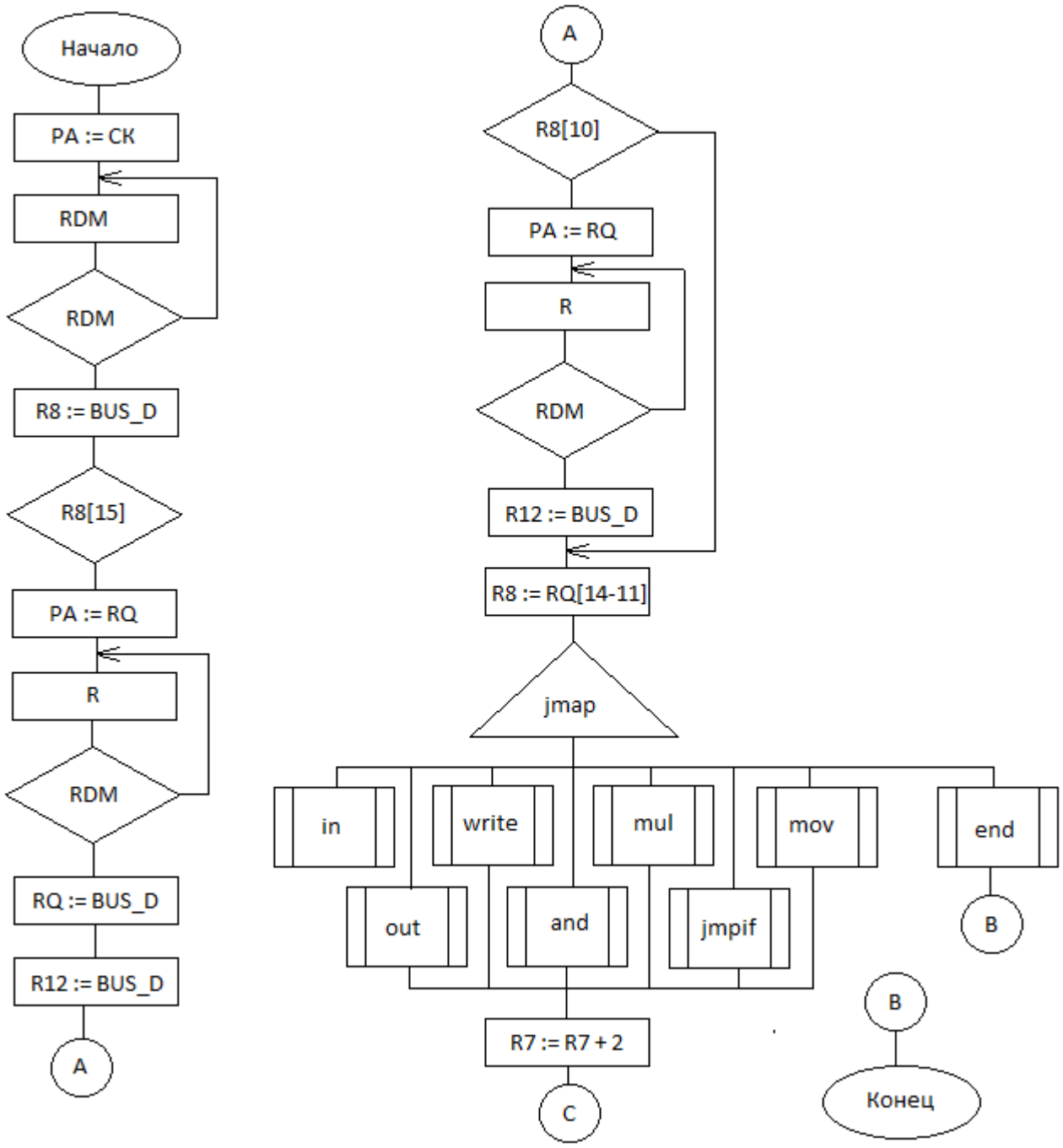
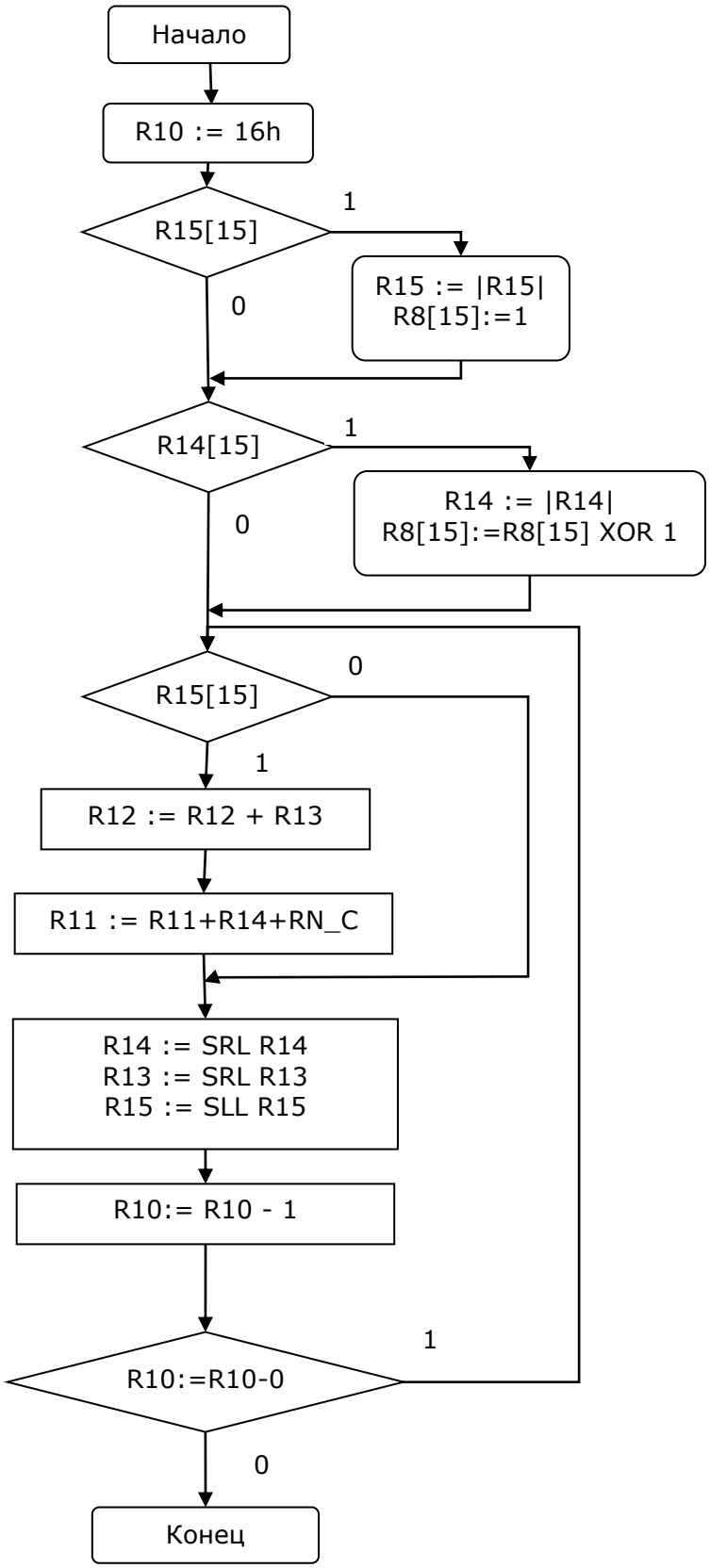


Рис 2.4 Схема функционального алгоритма распаковки, выборки и определения адреса следующей команды.

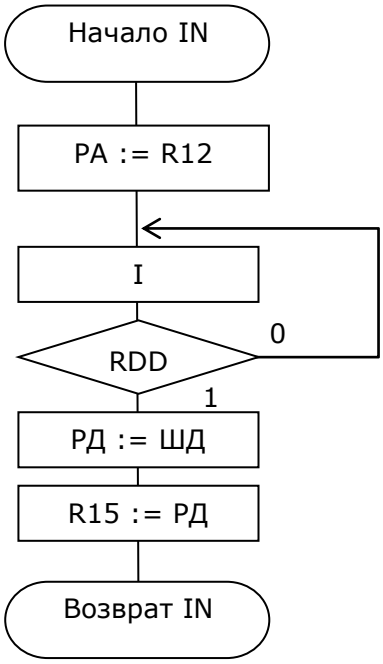
Далее микроалгоритмы рассмотренных выше операций.

1. Одноадресная команда умножения 4 способом. **MUL**

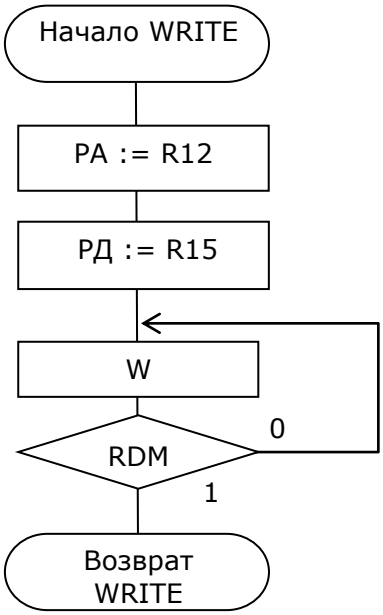
Операнды X – ПК, Y – ДК, результат – ДК.



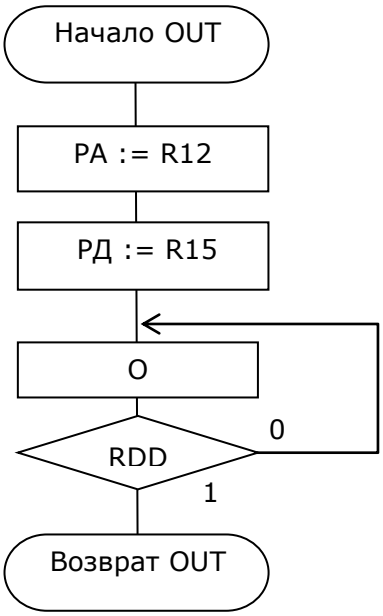
2. Микроалгоритм ввода с внешнего устройства, результат возвращается в аккумулятор. **IN**



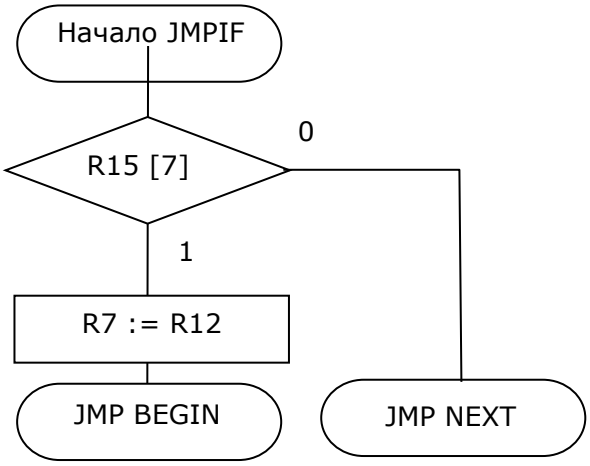
3. Одноадресная команда, записывающая содержимое аккумулятора в ОП по адресу хранящемуся в операнде. **WRITE**



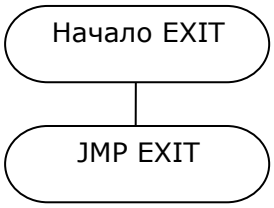
4. Одноадресная команда вывода на внешнее устройство содержимого аккумулятора. **OUT**



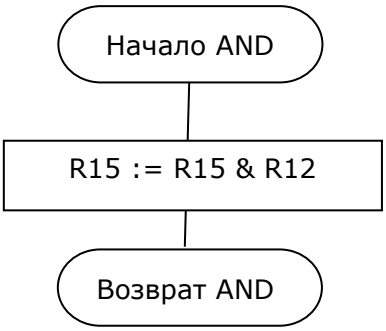
5. Одноадресная команда условного перехода по флагу, указывающему на то что в R15[7] = 0, т.е. R15 = 0080h. Если условие выполняется – переход на следующую команду, если не выполняется – на метку, чей адрес в R12. **JMPIF**



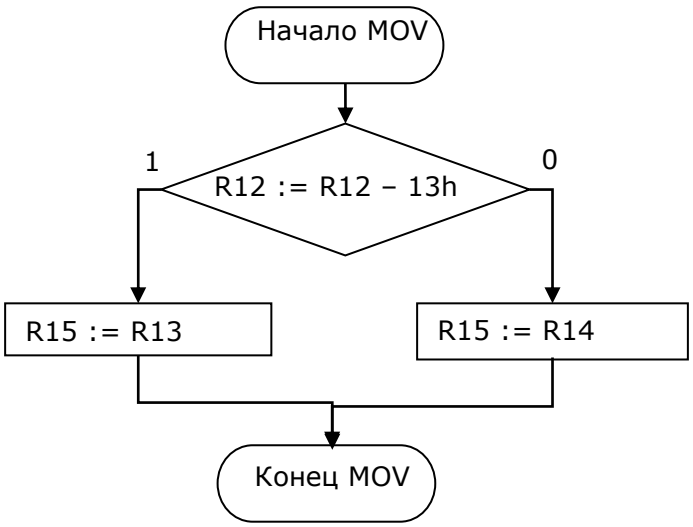
6. Одноадресная команда выхода из программы. Осуществляет переход по адресу выхода. **END**



7. Одноадресная команда конъюнкции. Операнды R15, R12, результат в R15. **AND**



8. Одноадресная команда перемещения содержимого одного из регистров (R13 или R14), чей номер находится в R12, в аккумулятор. **MOV**



3. Система команд и структура программного комплекса

При выборе системы команд особое внимание уделялось их универсальности. Если одноадресные команды работают с неявным операндом, то он хранится в аккумуляторе.

Одноадресные команды имеют следующий формат:

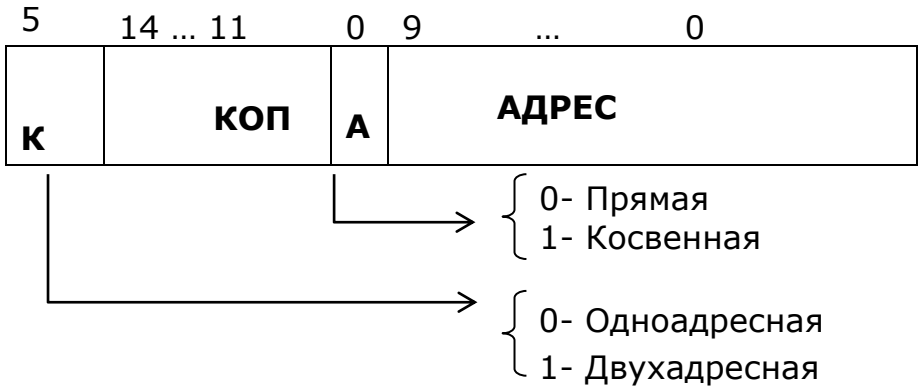


Рис 3.1 Формат одноадресных команд

Таблица 3.1 таблица кодов микрокоманд

Мнемоника	КОП	Описание
In	0001	Команда ввода
Out	0010	Команда вывода
Write	0011	Команда записи содержимого аккумулятора в память
And	0100	Команда конъюнкции содержимого аккумулятора и памяти
ul	0101	Команда умножения
JMmpif	0110	Условный переход по содержимому аккумулятора
End	0111	Команда завершения программы
Mov	1000	Команда пересылки содержимого R13/R14 -> R15

Структурная схема ЭВМ имеет вид:

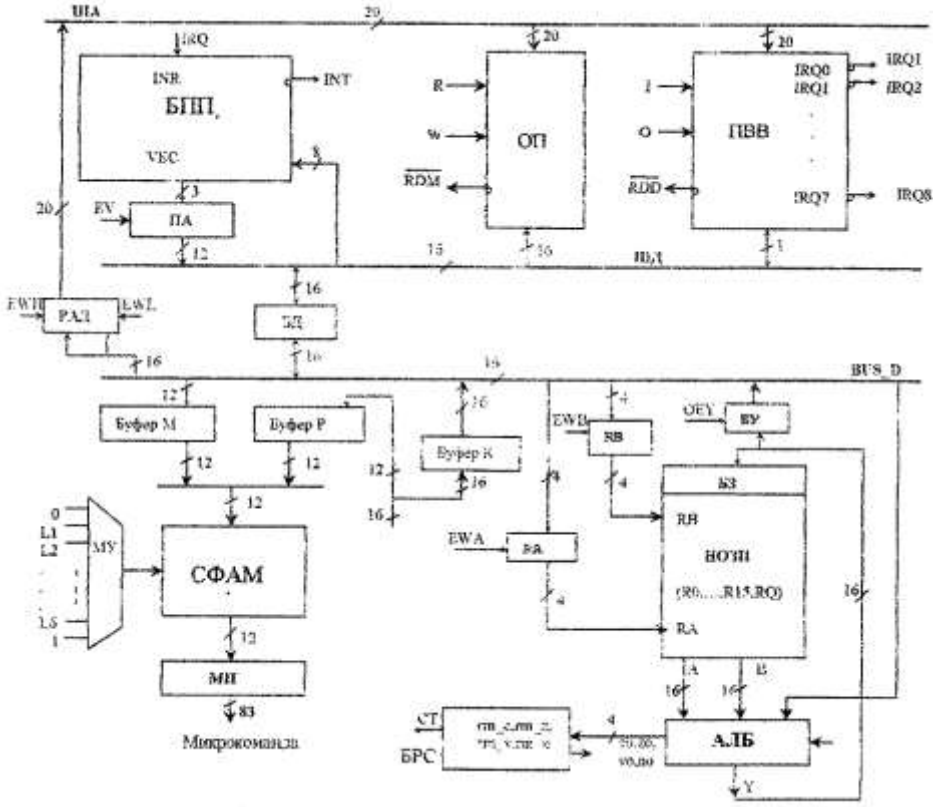


Рис. 3.2 Структурная схема комплекса 1804

4. Листинг программы

```
macro jmp label: {cjp nz, label; }

link l3:ct
link l1:rdm
link l2:rdd
link l4:irq
link l5:int
link ewh:16
link m: z, z, z, z, 14, 13, 12, 11, z, z, z, z

accept dev[3]: i, 12h, 14h, 2, 3
accept dev[4]: i, 4h, 6h, 2, 3
accept dev[6]: o, 16h, 18h, 2, 3

accept dev_buf[3]: 1111111111111110% \ X = -2
accept dev_buf[4]: 0B77h, 02AAh \ Y, Z

accept r7: 2h       \ Счетчик адресов команд

\ Программа
dw 0002h:0000100001001100% \ lb1: in RS dev3
dw 0004h:0011000000110101% \ if R15[0] jmp lb1
dw 0006h:0000100001001101% \ in r15, dev3 (X)
dw 0008h:0001100000110001% \ write X -> 31h
```

```
dw 000Ah:0000100001001010% \ lb2: in RS dev4
dw 000Ch:0011000000110110% \ if R15[0] jmp lb2
dw 000Eh:0000100001001011% \ in r15, dev4 (Y)
dw 0010h:0001100000110010% \ write Y -> 32h

dw 0012h:0000100001001010% \ lb3: in RS dev4
dw 0014h:0011000000110111% \ if R15[0] jmp lb3
dw 0016h:0000100001001011% \ in r15, dev4 (Z1)

dw 0018h:0010000000110010% \ and R15, Y (32h)
dw 001Ah:0010100000110001% \ mul r15, X (31h) -> rez r11, r12

dw 001Ch:0100000000110011% \ mov2 R13 -> R15
dw 001Eh:0001000001001111% \ out R15 -> dev(6)
dw 0020h:0100000000110100% \ mov2 R14 -> R15
dw 0022h:0001000001001111% \ out R15 -> dev(6)
dw 0024h:0011100000000000% \ exit
\ Конец программы

\ Переменные
dw 0031h:0031h \ X
dw 0032h:0032h \ Y
dw 0033h:0013h \ r13
dw 0034h:0014h \ r14

dw 0035h:0002h \ label lb1
dw 0036h:000Ah \ label lb2
dw 0037h:0012h \ label lb3

\ Устройства ввода-вывода
dw 004Ah:0004h \ addr RS dev4i
dw 004Bh:0006h \ addr RD dev4i
dw 004Ch:0012h \ addr RS dev3i
dw 004Dh:0014h \ addr RD dev3i
dw 004Eh:0016h \ addr RS dev6o
dw 004Fh:0018h \ addr RD dev6o

{cjp nz, begin; }

org 00010000% {jmp _in_; }        \ 0001
org 00100000% {jmp _out_; }       \ 0010
org 00110000% {jmp _write_; }     \ 0011
org 01000000% {jmp _and_; }       \ 0100
org 01010000% {jmp _mul_; }       \ 0101
org 01100000% {jmp _jmpif_; }     \ 0110
org 01110000% {jmp  exit; }       \ 0111
org 10000000% {jmp _mov_; }       \ 1000

begin
\ Выборка команды
{or nil, r7, z; ewl; oey; }
```

```
\ Считывание адреса команды в r8
{r; cjp rdm, cp; or r8, bus_d, z; }
\ Определение формата команды
{and rq, r8, 1000000000000000%; load rm, flags; }
{cjp not rm_z, twoadr; }       \ Двухадресная
{cjp nz, oneadr; }                    \ Одноадресная

twoadr {jmp next; }

oneadr {and rq, r8, 0000001111111111%; }
      \ Адрес операнда в rq и в PA
      {or rq, z; ewl; oey; }
     \ Считывание операнда в r12
      {r; cjp rdm, cp; or r12, bus_d, z; }
     \ Проверка типа адресации
      {and rq, r8, 0000010000000000%; load rm, flags; }
      {cjp rm_z, pryamaya; } \ Если прямая, выполнить прыжок
      \ Иначе
      {or r12, z; ewl; oey; }
      \ Заносим операнд в r12
      {r; cjp rdm, cp; or r12, bus_d, z; }

pryamaya {and rq, r8, 0111100000000000%; } \ Проверка КОПа
             {jmap; or rq, z; oey; }              \ Переход по КОПу


\ Умножение
\ X - R15, Y - R12
\ Результат в R13, R14
_mul_  {or r10, 16; } \ Счетчик
      {or r14, r12, z; }
      {xor r11, r11, r11; }
      {xor r12, r12, r12; }
      {xor r13, r13, r13; }

      \ Проверка знака в r15
      {or r8, r15, z; cjp not no, label1; }
      {sub r15, z, r15, nz; }
      \ Запоминание знака
label1 {xor r8, r14; }
      {and r8, r8, 8000h; }
      {and r15, r15, 7fffh; }

      \ Проверка знака в r14
      {or r9, r14, z; cjp not no, label11; }
      {sub r14, z, r14, nz; }
      {and r14, r14, 7fffh; }

      \ Умножение
label11 {or srl, r14, r14, z; }
       {or sr.9, r13, r13, z; }
```

```
        {or sll, r15, r15; }
        {cjp not rm_c, label2; }
        {add r12, r12, r13; }
        {add r11, r11, r14, rn_c; }
label2  {sub r10, r10, z, z; load rm, flags; cem_c; }
        {cjp not rm_z, label11; }
        {or r8, z; cjp not no, label3; }
        {sub r12, z, r12, nz; load rm, flags; }
        {sub r11, z, r11, rm_c; }
label3  {or r11, r8; }
        {xor r13, r13, r13; }
        {xor r14, r14, r14; }
        {or r13, r11, z; }
        {or r14, r12, z; }
        {xor r11, r11, r11; }
        {xor r12, r12, r12; }
        {jmp next; }


\ Команда ввода с устройства в R15
_in_    {or r12, z; ewl; oey; }
        {i; cjp rdd, cp; or r15, bus_d, z; }
        {cjp nz, next; }

\ Команда вывода в устройство из R15
_out_   {or nil, r12, z; ewl; oey; }
        {or nil, r15, z; oey; o; cjp rdd, cp; }
        {jmp next; }

\ Запись из R15 в память по адресу R12
_write_{or nil, r12, z; ewl; oey; }
       {w; or nil, r15, z; oey; cjp rdm, cp; }
       {jmp next; }

\ Конъюнкция содержимого R15 и R12
_and_   {and r15, r15, r12; }
        {jmp next; }

\ Условный переход на метку по содержимому R15
_jmpif_ {and nil, r15, 10000000%; load rm, flags; }
        {cjp not rm_z, next; }
        {or r7, r12, z; }
        {jmp begin; }

\ Перемещение из R13 или R14 в R15
_mov_ {sub nil, r12, 13h, nz; load rm, flags; }
       {cjp not rm_z, lr14; }
       {or r15, r13, z; }
       {jmp next; }
lr14   {or r15, r14, z; }
       {jmp next; }
```

```
\ Формирование адреса следующей команды
next    {add r7, r7, 2, z; }
        {cjp nz, begin; }

\ Завершение команды
exit    {}
```

5. Программа на ассемблере

```
lb1:   in R15, RS dev3   ; читаем X
if R15[0] jmp lb1
in R15, RD dev3
write 0031h, R15         ; сохраняем в памяти X

lb2:   in R15, RS dev4   ; читаем Y
if R15[0] jmp lb2
in R15, RD dev4
write 0032h, R15         ; сохраняем в памяти Y

lb3:   in R15, RS dev4   ; читаем Z
if R15[0] jmp lb3
in R15, dev4

and R15, 0032h           ; R15 := Y&Z
mul r15, 0031h           ; R11, R12 := X*R15
mov R15, R13             ; R13 -> R15
out R15, dev(6)          ; вывод верхней части результата
mov R15, R14             ; R14 -> R15
out R15, dev(6)          ; вывод нижней части результата
exit
```

6. Вывод

В ходе написания этой контрольной была разработана программа, реализующая вычисления функции A = X * (Z & Y). Данные вводились и выводились посредством внешних устройств. На микропрограммном уровне была разработана система команд и реализован алгоритм организации процесса вычислений.

В процессе разработки стало очевидно, что чем проще программа на ассемблере, тем сложнее должна быть её реализация на микропрограммном уровне. Для каждой команды должна быть заранее разработана её реализация при помощи микроопераций. Благодаря этому сокращается длительность и количество тактов основной программы.