Санкт-Петербургский политехнический университет Петра Великого

Институт информационных технологий и управления

Кафедра «Информационные и управляющие системы»

**Курсовая работа**

**Разработка учебной системы программирования**

**Компилятор с языка Assembler.**

по дисциплине «Системы программирования»

Вариант 9

Выполнили

студенты гр. 53504/3

Мамонтов Я. С.

Кузнецов Д. А.

Хутар Давуд Захи

Руководитель:

Расторгуев В. Я.

**«\_\_\_» \_\_\_\_\_\_\_\_\_\_** 2016 г.

Санкт-Петербург

2016

# Введение

Данная курсовая работа имеет своей целью получение практических навыков построения компилятора с языка Ассемблер, являющегося одним из элементов системы программирования, образующих в совокупности следующий технологический конвейер:

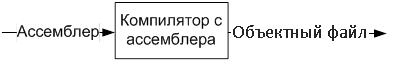


При этом предполагается то, что данная система программирования работает на технологической ЭВМ (IBM PC) и является по существу кросс-системой для объектной ЭВМ (ЕС ЭВМ). В этой системе:

* в качестве языка высокого уровня (ЯВУ) выбран язык, образованный из подмножества языковых конструкций ПЛ1, а исходная программа готовится в виде текстового файла технологической ЭВМ с расширением \*.pli;
* язык АССЕМБЛЕРА сформирован из языковых конструкций АССЕМБЛЕРА ЕС ЭВМ, а ассемблеровский эквивалент исходной программы формируется в виде текстового файла технологической ЭВМ с расширением \*.ass;
* объектный эквивалент исходной программы готовится в формате объектных файлов операционной системы ОС ЕС ЭВМ и хранится в виде двоичного файла технологической ЭВМ с расширением \*.tex;
* загрузочный эквивалент исходной программы представляет собой машинный код ЕС ЭВМ, запоминаемый в области ОЗУ технологической ЭВМ, являющейся зоной загрузки для эмулятора объектной ЭВМ.

# Постановка задачи

Необходимо выполнить доработку элементов макета учебной системы программирования до уровня, позволяющего обрабатывать “новые” для макета конструкции языка высокого уровня, примененные в соответствующем варианте:



На входе имеется исходный код программы на ассемблере ЭВМ IBM 370:

|  |
| --- |
| EX09 START 0 Start of the programm  BALR @RBASE,0 Load the register of the base  USING \*,@RBASE Set register as the base  L @RRAB,A Load the variable to the register  SRL @RRAB,29 Logic shift  CVD @RRAB,@BUF Change type  LA @RADD,@BUF Load addres to register  MVC B(3),5(@RADD)Moving to needed variable  BCR 15,14 Exit from the programm  A DC BL4'101' Definition of variable  B DC PL3'0' Definition of variable  DS 0F Aligment  @BUF DC PL8'0' Definition of variable  @RBASE EQU 15  @RRAB EQU 5  @RADD EQU 4  END End of the programm |

В результате работы компилятора должен получится эквивалент исходной программы в виде объектного файла для IBM 370:

|  |
| --- |
| 0245 5344 4040 4040 4040 0010 4040 0001  4558 3039 4040 4040 0000 0000 4000 002c  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 0000 4040 0002 4040 0001  05f0 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 0002 4040 0004 4040 0001  5850 f018 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 0006 4040 0004 4040 0001  8850 001d 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 000a 4040 0004 4040 0001  4e50 f022 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 000e 4040 0004 4040 0001  4140 f022 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 0012 4040 0006 4040 0001  d203 f01c 4005 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 0018 4040 0002 4040 0001  07fe 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 001a 4040 0004 4040 0001  a000 0000 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 001e 4040 0003 4040 0001  0000 0040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0254 5854 4000 0024 4040 0008 4040 0001  0000 0000 0000 0000 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040  0245 4e44 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4040 4040 4040 4040  4040 4040 4040 4040 4558 3039 4040 4040 |

# Анализ поставленной задачи

Команды ассемблера кодируются следующими кодами:

|  |  |
| --- | --- |
| SRL @RRAB, 13  CVD @RRAB, @BUF  LA @RADD, @BUF  MVC B(3), 5(@RADD) | **88 50 00 0d**  **4e 50 f0 46**  **41 40 f0 46**  **d2 03 f0 1c 40 05** |

# Входные ограничения

Битовая строка должна быть не более 1 байта.

# Модификация базы данных исходного макета

Таблица машинных операций имеет вид:

|  |
| --- |
| // Кодирование дополнительных команд для нашего варианта  **{{'L','A',' ',' ',' '} , '\x41' , 4 , FRX} ,**  **{{'C','V','D',' ',' '} , '\x4E' , 4 , FRX} ,**  **{{'S','R','L',' ',' '} , '\x88' , 4 , FRS} ,**  **{{'M','V','C',' ',' '} , '\xD2' , 6 , FSS}** |

Также в начале второго просмотра были установлены указатели на программные обработчики новых команд:

|  |
| --- |
| // Указатели на подпрограммы-обработчики команд АССЕМБЛЕРА  // при втором просмотре  **T\_MOP[6].BXPROG = SRX;**  **T\_MOP[7].BXPROG = SRX;**  **T\_MOP[8].BXPROG = SRS;**  **T\_MOP[9].BXPROG = SSS;** |

# Модификация алгоритма исходного макета

С целью расширения функциональности языка в функции компилятора были внесены изменения, представленные ниже:

//БЛОК объявлений подпрограмм, используемых при 1-ом проходе

|  |
| --- |
| **int FDC()**  **{**  **int size;**  **if ( PRNMET == 'Y' )**  **{**  **if (TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[0]=='F')**    **{**  **T\_SYM[ITSYM].DLSYM = 4;**  **T\_SYM[ITSYM].PRPER = 'R';**  **if ( CHADR % 4 )**  **{**  **CHADR = (CHADR /4 + 1) \* 4;**  **T\_SYM[ITSYM].ZNSYM = CHADR;**  **}**  **PRNMET = 'N';**  **}**  **else if (TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[0]=='B')**  **{**  **if (TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[1]=='L')**  **{**  **size = atoi(&TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[2]);**  **T\_SYM[ITSYM].DLSYM = size;**  **T\_SYM[ITSYM].PRPER = 'R';**  **T\_SYM[ITSYM].ZNSYM = CHADR;**  **PRNMET = 'N';**  **}**  **}**  **else if (TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[0]=='P')**  **{**  **if (TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[1]=='L')**  **{**  **size = atoi(&TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[2]);**  **T\_SYM[ITSYM].DLSYM = size;**  **T\_SYM[ITSYM].PRPER = 'R';**  **T\_SYM[ITSYM].ZNSYM = CHADR;**  **PRNMET = 'N';**  **}**  **}**  **else**  **return (1); /\* иначе выход по ошибке \*/**  **}**  **else /\*если же псевдооп.непомеч\*/**  **if ( CHADR % 4 ) /\*и CHADR не кратен 4,то: \*/**  **CHADR = (CHADR /4 + 1) \* 4; /\* установ.CHADR на гр.сл.\*/**  **PRNMET = 'N';**  **CHADR = CHADR + size; /\*увелич.CHADR на 4 и \*/**  **return (0); /\*успешно завершить подпр.\*/**  **}** |
| **int FDS()**  **{**  **if ( PRNMET == 'Y' )**  **{**  **if (TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[1]=='F')**        **{**  **T\_SYM[ITSYM].DLSYM = 4;**  **T\_SYM[ITSYM].PRPER = 'R';**  **if ( CHADR % 4 )**  **{**  **CHADR = (CHADR /4 + 1) \* 4;**  **}**  **else**  **{**  **CHADR = CHADR + 4;**  **}**  **PRNMET = 'N';**  **}**  **else**  **{**  **return (1);**  **}**  **}**  **else**  **{**  **T\_SYM[ITSYM].DLSYM = 4;**  **T\_SYM[ITSYM].PRPER = 'R';**  **if ( CHADR % 4 )**  **CHADR = (CHADR /4 + 1) \* 4;**  **else**  **CHADR = CHADR + 4;**  **}**  **T\_SYM[ITSYM].ZNSYM = CHADR;**  **return (0);**  **}** |
| **//Обработка RS комманд на первом проходе**  **int FRS()**  **{**  **CHADR = CHADR + 4;**  **if ( PRNMET == 'Y' )**  **{**  **T\_SYM[ITSYM].DLSYM = 4;**  **T\_SYM[ITSYM].PRPER = 'R';**  **}**  **return(0);**  **}** |
| **//Обработка SS комманд на первом проходе**  **int FSS()**  **{**  **CHADR = CHADR + 6;**  **if ( PRNMET == 'Y' )**  **{**  **T\_SYM[ITSYM].DLSYM = 6;**  **T\_SYM[ITSYM].PRPER = 'R';**  **}**  **return 0;**  **}** |

// Второй проход

|  |
| --- |
| **int SDC()**  **{**  **char \*RAB; /\*рабочая переменная \*/**  **RX.OP\_RX.OP = 0; /\*занулим два старших \*/**  **RX.OP\_RX.R1X2 = 0; /\*байта RX.OP\_RX \*/**  **if (!memcmp(TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND, "F'", 2))**  **{**  **RAB = strtok((char\*)TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND+2, "'");**  **RX.OP\_RX.B2D2 = atoi ( RAB ); /\*перевод ASCII-> int \*/**  **RAB = (char \*) &RX.OP\_RX.B2D2; /\*приведение к соглашениям\*/**  **swab ( RAB , RAB , 2 ); /\* ЕС ЭВМ \*/**  **STXT (4); /\*формирование TXT-карты \*/**  **return (0); /\*успешн.завершение подпр.\*/**  **}**  **else if ( !memcmp( TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND, "BL", 2 ) )**  **{**  **RAB=strtok( (char\*)TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND + 4, "'" );**  **//Our awesome input**  **int size = strlen(RAB);**  **int value = strtol( RAB, NULL, 2 );**  **char buffer[1];**  **buffer[0] = value<<(8-size);**  **memcpy(BL\_BUFFER, buffer, 1);**  **STXT (1); /\*формирование TXT-карты \*/**  **return (0);**  **}**  **else if ( !memcmp(TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND, "PL", 2) )**  **{**  **RAB=strtok( (char\*)TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND+4, "'" );**  **int size = TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[2]-'0';**    **RX.OP\_RX.B2D2 = atoi ( RAB ); /\*перевод ASCII-> int \*/**  **RAB = (char \*) &RX.OP\_RX.B2D2; /\*приведение к соглашениям\*/**  **printf("1 -> %d\n", RX.OP\_RX.B2D2);**  **printf("1 -> %s\n", RAB);**  **char buffer[8];**  **memset ( buffer , 64 , 8 );**  **memset ( buffer , 0 , size-1 );**  **buffer[size-1] = RX.OP\_RX.B2D2;**  **if (RX.OP\_RX.B2D2 >= 0)**  **{**  **buffer[0] = 0xc<<4;**  **}**  **if (size <= 4)**  **{**  **memcpy(RX.BUF\_OP\_RX, buffer, 4);**  **printf("2 -> %s\n", RX.BUF\_OP\_RX);**  **STXT (size);**  **}**  **else**  **{**  **memcpy(PL8\_BUFFER, buffer, 8);**  **STXT (size);**  **}**  **return (0);**  **}**  **else**  **return (1);**  **}** |
| **int SDS() /\*подпр.обр.пс.опер.DS \*/**  **{**  **RX.OP\_RX.OP = 0; /\*занулим два старших \*/**  **RX.OP\_RX.R1X2 = 0; /\*байта RX.OP\_RX \*/**  **if ( TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND[1]=='F')**  **{ /\* то: \*/**  **RX.OP\_RX.B2D2 = 0; /\*занулим RX.OP\_RX.B2D2 \*/**  **}**  **else /\*иначе \*/**  **return (1); /\*сообщение об ошибке \*/**  **if ( CHADR % 4 )**  **{ /\* если не кратен 4 \*/**  **STXT2(4 - (CHADR % 4)); /\* делаем выравнивание\*/**  **}**  **else**  **{**  **STXT2(4);**  **}**  **return (0);**  **}** |
| **int SRS()**  **{**  **char \*METKA; /\*набор \*/**  **char \*METKA1; /\*рабочих \*/**  **char \*METKA2; /\*переменных \*/**  **char \*PTR; /\* \*/**  **int DELTA; /\* \*/**  **int ZNSYM; /\* \*/**  **int NBASRG; /\* \*/**  **int J; /\* \*/**  **int I; /\* \*/**  **unsigned char R1X2; /\* \*/**  **int B2D2;**  **int DLSYM; /\* \*/**  **RX.OP\_RX.OP = T\_MOP[I3].CODOP; /\*формирование кода операц\*/**  **METKA1 = strtok((char\*) TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND, ",");**  **METKA2 = strtok(NULL, " ");**  **if ( isalpha ( (int) \*METKA1 ) || METKA1[0] == '@' )**  **{**  **for ( J=0; J<=ITSYM; J++ )**  **{**  **METKA = strtok((char\*) T\_SYM[J].IMSYM , " ");**  **if(!strcmp (METKA, METKA1))**  **{**  **NBASRG = 0;**  **DELTA = 0xfff - 1;**  **ZNSYM = T\_SYM[J].ZNSYM;**  **DLSYM = T\_SYM[J].DLSYM; /\* смещен.втор.операнда \*/**  **R1X2 = T\_SYM[J].ZNSYM << 4;**  **// New code**  **DELTA = atoi(METKA2);**  **B2D2 = NBASRG << 12;**  **B2D2 = B2D2 + DELTA;**  **PTR = (char \*)&B2D2;**  **swab ( PTR , PTR , 2 );**  **RX.OP\_RX.B2D2 = B2D2;**  **RX.OP\_RX.R1X2 = R1X2;**  **STXT(4);**  **return(0);**  **}**  **}**  **return(2);**  **}**  **else**  **{**  **R1X2 = atoi ( METKA1 ) << 4;**  **}**  **}** |
| **int SSS() {**  **unsigned i, j;**  **unsigned rbase, delta, offset;**  **char \*op1, \*len1, \*op2, \*op3;**  **char \*tmp;**  **op1 = strtok((char \*) TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND, "(");**  **len1 = strtok(NULL, ")");**  **char \* t = strtok(NULL, ",");**  **op2 = strtok(t, "(");**  **op3 = strtok(NULL, ")");**  **SS.OP\_SS.OP = T\_MOP[I3].CODOP;**  **SS.OP\_SS.L1 = atoi(len1);**  **if (isalpha((int) \*op1) || op1[0] == '@')**  **{**  **for (i = 0; i <= ITSYM; i++)**  **{**  **tmp = strtok((char \*) T\_SYM[i].IMSYM, " ");**  **if (!strcmp(tmp, op1))**  **{**  **rbase = 0;**  **delta = 0xfff - 1;**  **offset = T\_SYM[i].ZNSYM;**  **for (j = 0; j < 15; j++)**  **{**  **if (T\_BASR[j].PRDOST == 'Y' &&**  **offset - T\_BASR[j].SMESH >= 0 &&**  **offset - T\_BASR[j].SMESH < delta)**  **{**  **rbase = j + 1;**  **delta = offset - T\_BASR[j].SMESH;**  **}**  **}**  **if (rbase == 0 || delta > 0xfff)**  **return 5;**  **else**  **{**  **SS.OP\_SS.B1D1 = rbase << 12;**  **SS.OP\_SS.B1D1 = SS.OP\_SS.B1D1 + delta;**  **tmp = (char \*) &SS.OP\_SS.B1D1;**  **swab(tmp, tmp, 2);**  **goto CNT1;**  **}**  **}**  **}**  **printf("FAIL 1\n");**  **return 2;**  **}**  **else**  **printf("FAIL 2\n");**  **return 2;**  **CNT1:**  **if (isalpha((int) \*op3) || op3[0] == '@')**  **{**  **for (i = 0; i <= ITSYM; i++)**  **{**  **tmp = strtok((char \*) T\_SYM[i].IMSYM, " ");**  **if (!strcmp(tmp, op3))**  **{**  **SS.OP\_SS.X2 = atoi(op2);**  **SS.OP\_SS.L2 = T\_SYM[i].ZNSYM << 4;**  **goto CNT2;**  **}**  **}**  **}**  **CNT2:**  **STXT(6);**    **return 0;**  **}** |
| **void STXT( int ARG ) /\*подпр.формир.TXT-карты \*/**  **{**  **char \*PTR; /\*рабоч.переменная-указат.\*/**  **PTR = (char \*)&CHADR; /\*формирование поля ADOP \*/**  **TXT.STR\_TXT.ADOP[2] = \*PTR; /\*TXT-карты в формате \*/**  **TXT.STR\_TXT.ADOP[1] = \*(PTR+1); /\*двоичного целого \*/**  **TXT.STR\_TXT.ADOP[0] = '\x00'; /\*в соглашениях ЕС ЭВМ \*/**  **if ( ARG == 1 ) /\*формирование поля OPER \*/**  **{**  **memset ( TXT.STR\_TXT.OPER , 64 , 8 );**  **memcpy ( TXT.STR\_TXT.OPER,BL\_BUFFER , 4 );**  **TXT.STR\_TXT.DLNOP [1] = 4;**  **ARG = 4;**  **}**  **else if ( ARG == 2 )**  **{**  **memset ( TXT.STR\_TXT.OPER , 64 , 8 );**  **memcpy ( TXT.STR\_TXT.OPER,RR.BUF\_OP\_RR , 2 ); /\* для RR-формата \*/**  **TXT.STR\_TXT.DLNOP [1] = 2;**  **}**  **else if (ARG == 4)**  **{**  **memcpy ( TXT.STR\_TXT.OPER , RX.BUF\_OP\_RX , 4);/\* для RX-формата \*/**  **TXT.STR\_TXT.DLNOP [1] = 4;**  **}**  **else if (ARG == 6)**  **{**  **memcpy ( TXT.STR\_TXT.OPER , SS.BUF\_OP\_SS , 6);/\* для SS-формата \*/**  **TXT.STR\_TXT.DLNOP [1] = 6;**  **}**  **else if (ARG ==8)**  **{**  **memset ( TXT.STR\_TXT.OPER , 64 , 8 );**  **memcpy ( TXT.STR\_TXT.OPER , PL8\_BUFFER , 8); /\* для PL8 \*/**  **TXT.STR\_TXT.DLNOP [1] = 8;**  **}**  **else**  **{**  **memset ( TXT.STR\_TXT.OPER , 64 , 8 );**  **memcpy ( TXT.STR\_TXT.OPER , RX.BUF\_OP\_RX , ARG);/\* для PL \*/**  **TXT.STR\_TXT.DLNOP [1] = ARG;**  **}**  **memcpy (TXT.STR\_TXT.POLE9,ESD.STR\_ESD.POLE11,8);/\*формиров.идентифик.поля \*/**  **memcpy ( OBJTEXT[ITCARD] , TXT.BUF\_TXT , 80 ); /\*запись об'ектной карты \*/**  **ITCARD += 1; /\*коррекц.инд-са своб.к-ты\*/**  **CHADR = CHADR + ARG; /\*коррекц.счетчика адреса \*/**  **return;**  **}**  **// Используется SDS для выравнивания адреса заолнением памяти до значения кратного 4**  **void STXT2( int ARG ) /\*подпр.формир.TXT-карты \*/**  **{**  **char \*PTR; /\*рабоч.переменная-указат.\*/**  **PTR = (char \*)&CHADR; /\*формирование поля ADOP \*/**  **TXT.STR\_TXT.ADOP[2] = \*PTR; /\*TXT-карты в формате \*/**  **TXT.STR\_TXT.ADOP[1] = \*(PTR+1); /\*двоичного целого \*/**  **TXT.STR\_TXT.ADOP[0] = '\x00'; /\*в соглашениях ЕС ЭВМ \*/**  **memcpy ( TXT.STR\_TXT.OPER , RX.BUF\_OP\_RX , 0);/\* для RX-формата \*/**  **TXT.STR\_TXT.DLNOP [1] = ARG;**  **memcpy (TXT.STR\_TXT.POLE9,ESD.STR\_ESD.POLE11,8);/\*формиров.идентифик.поля \*/**  **memcpy ( OBJTEXT[ITCARD] , TXT.BUF\_TXT , 80 ); /\*запись об'ектной карты \*/**  **ITCARD += 1; /\*коррекц.инд-са своб.к-ты\*/**  **CHADR = CHADR + ARG; /\*коррекц.счетчика адреса \*/**  **return;**  **}** |

# Заключение

В ходе выполнения данной части курсовой работы в алгоритм компилятора с ассемблера были внесены изменения, позволяющие использовать в коде обрабатываемой им программы языковые конструкции, представленные в задании. В результате данной части курсовой работы был получен объектный модуль. Проверить его правильность можно с помощью абсолютного загрузчика и эмулятора машины.

Полученный в ходе работы компилятор был успешно скомпилирован. После его запуска с поданным на вход кодом ассемблера, полученным в результате работы компилятора с языка PL/1, примененного к коду варианта задания № 9, был получен объектный файл для IBM 370, который полностью соответствует разработанному вручную варианту, описанному в главе «Постановка задачи».