Минобрнауки России

Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра «Информационные и управляющие системы»

КУРСОВОЙ ПРОЕКТ

Разработка учебной системы программирования

Построение компилятора с Ассемблера

по дисциплине «Системы программирования»

Выполнили

студенты гр.5084/12 А.А.Лукашин

К.С.Шубин

Руководитель

доцент В.Я.Расторгуев

«__» _____ 2013 г.

Санкт-Петербург

2013

Оглавление

Задание

Воспользовавшись результатами первого этапа курсовой работы, доработать существующий компилятор с ассемблера для получения объектного представления программы. В дальнейшем результат будет использован в третьей части курсовой работы.

Вариант №7:

Код на языке PL1

```
EX07: PROC OPTIONS (MAIN);

            DCL A BIT (3) INIT ( 10B );

            DCL B BIT (3) INIT ( 101B );

            DCL C BIT (16);

            C = SUBSTR((B !! A),2,3);

END EX07;
```

В результате выполнения первой части курсовой работы был получен эквивалент программы на языке Ассемблер.

```
EX07     START 0            Начало программы
         BALR  RBASE,0      Загрузка регистра базы
         USING *,RBASE      Назначить регистр базой
         LH    3,B          Загрузка переменной B в регистр
         LH    4,A          Загрузка переменной A в регистр
         OR    3,4          Логическое «ИЛИ» регистров
         SRL   4,3          Сдвиг операнда вправо
         SLL   3,2          Сдвиг операнда влево
         LH    4,TMP        Загрузка маски в регистр
         NR    3,4          Логическое «И» регистров
         STH   3,C          Формирование результата
         BCR   15,RVIX      Выход из программы
A        DS    0H           Выравнивание адреса
         DC    BL2'10'      Инициализация переменной
B        DS    0H           Выравнивание адреса
         DC    BL2'101'     Инициализация переменной
C        DS    0H           Выравнивание адреса
         DS    BL2          Объявление без инициализации
TMP      DS    0H           Выравнивание адреса
         DC    BL2'111'     Инициализация переменной
RBASE    EQU   5
RVIX     EQU   14
         END   EX07         Конец программы
```

План работы

Необходимо доработать компилятор с Ассемблера в объектное представление, дополнив его новой функциональностью. В новую функциональность входят:

1. поддержка новых команд (LH, STH, SRL, SLL, OR, NR);
2. поддержка типа BL2.

Для решения этой задачи необходимо:

1. расширить таблицу машинных команд;
2. модифицировать обработчики команд DC и DS первого и второго просмотров, а также изменить обработчик команд RX-типа.

Расширение таблицы команд

```
#define DL_ASSTEXT 29
#define DL_OBJTEXT 100    /*длина об'ектн. текста   */
#define NSYM 50           /*размер табл.символов    */
#define NPOP 6            /*размер табл.псевдоопер. */
#define NOP  12           /*размер табл.операций    */
```

Таблица машинных операций имеет следующий вид (добавленные фрагменты выделены цветом):

```
T_MOP [NOP]  =
    {
     {{'B','A','L','R',' '} , '\x05' , 2 , FRR} ,
     {{'B','C','R',' ',' '} , '\x07' , 2 , FRR} ,
     {{'S','T',' ',' ',' '} , '\x50' , 4 , FRX} ,
     {{'L',' ',' ',' ',' '} , '\x58' , 4 , FRX} ,
     {{'A',' ',' ',' ',' '} , '\x5A' , 4 , FRX} ,
     {{'S',' ',' ',' ',' '} , '\x5B' , 4 , FRX} ,
     {{'L','H',' ',' ',' '} , '\x48' , 4 , FRX} ,
     {{'S','R','L',' ',' '} , '\x01' , 4 , FRX} ,
     {{'S','L','L',' ',' '} , '\x02' , 4 , FRX} ,
     {{'O','R',' ',' ',' '} , '\x16' , 4 , FRR} ,
     {{'N','R',' ',' ',' '} , '\x14' , 4 , FRR} ,
     {{'S','T','H',' ',' '} , '\x40' , 4 , FRX} ,
    };
```

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

```
CONT3:
 T_MOP[0].BXPROG = SRR;
```

```
T_MOP[1].BXPROG = SRR;
T_MOP[2].BXPROG = SRX;
T_MOP[3].BXPROG = SRX;
T_MOP[4].BXPROG = SRX;
T_MOP[5].BXPROG = SRX;
T_MOP[6].BXPROG = SRX;
T_MOP[7].BXPROG = SRX;
T_MOP[8].BXPROG = SRX;
T_MOP[9].BXPROG = SRR;
T_MOP[10].BXPROG = SRR;
T_MOP[11].BXPROG = SRX;
```

Модификация кода компилятора

В данном разделе рассмотрены модифицированные функции компилятора с ассемблера.

Первый просмотр:

1) Определение оператора DC при первом просмотре

```
int FDC()
{
	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] == 'H') {
			T_SYM[ITSYM].DLSYM = 2;
			T_SYM[ITSYM].PRPER = 'R';
			if (CHADR % 2)
			{
				CHADR = (CHADR / 2 + 1) * 2;
				T_SYM[ITSYM].ZNSYM = CHADR;
			}
			PRNMET = 'N';
		} else if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == 'B'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'L'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[2] == '2')
		{
			T_SYM[ITSYM].DLSYM = 2;
			T_SYM[ITSYM].PRPER = 'R';
			if (CHADR % 2)
			{
				CHADR = (CHADR / 2 + 1) * 2;
				T_SYM[ITSYM].ZNSYM = CHADR;
			} else {
				CHADR += 2;
			}
			PRNMET = 'N';
		} else
			return (1);
```

```
		return 0;
	} else {
		if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == 'B'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'L'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[2] == '2')
		{
			T_SYM[ITSYM].DLSYM = 2;
			T_SYM[ITSYM].PRPER = 'R'
			if (CHADR % 2) /*  и, если CHADR не указ.*/
			{
				CHADR = (CHADR / 2 + 1) * 2;
				T_SYM[ITSYM].ZNSYM = CHADR;
			} else {
				CHADR += 2;
			}
			PRNMET = 'N';
		}
		return (0);
	}
}
```

2) Определение оператора DS при первом просмотре

```
int FDS
{
	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;
			}
			CHADR += 4;
			PRNMET = 'N';
		} else if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == 'B'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'L'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[2] == '2')
		{
			T_SYM[ITSYM].DLSYM = 2;
			T_SYM[ITSYM].PRPER = 'R'
			if (CHADR % 2)
			{
				CHADR = (CHADR / 2 + 1) * 2;
			} else {
				CHADR += 2;
			}
			PRNMET = 'N';
		} else if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == '0'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'H')
		{
			T_SYM[ITSYM].DLSYM = 2;
			T_SYM[ITSYM].PRPER = 'R';
			if (CHADR % 2)
			{
				CHADR = (CHADR / 2 + 1) * 2;
				T_SYM[ITSYM].ZNSYM = CHADR;
			}
			PRNMET = 'N';
		} else
			return (1);
		return 0;
	} else {
		if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == 'B'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'L'
			&& TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[2] == '2')
	{
			T_SYM[ITSYM].DLSYM = 2;
```

```
                        T_SYM[ITSYM].PRPER = 'R';
                        if (CHADR % 2)
                        {
                                CHADR = (CHADR / 2 + 1) * 2;
                        } else {
                                CHADR += 2;
                        }
                        PRNMET = 'N';
                } else if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == '0'
                        && TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'H')
                {
                        T_SYM[ITSYM].DLSYM = 2;
                        T_SYM[ITSYM].PRPER = 'R';
                        if (CHADR % 2)
                        {
                                CHADR = (CHADR / 2 + 1) * 2;
                                T_SYM[ITSYM].ZNSYM = CHADR;
                        }
                        PRNMET = 'N';
                } else
                        return (1);
        }
        return (0);
}
```

3) Определение операции RX при первом просмотре

```
 int FRX()
{
        CHADR = CHADR + 4;
        if (PRNMET == 'Y')
        {
                T_SYM[ITSYM].DLSYM = 4;
                T_SYM[ITSYM].PRPER = 'R';
        }
        return (0);
}
```

Второй просмотр:

1) Определение оператора DC при втором просмотре

```
int SDC()
{
        char *RAB;
        RX.OP_RX.OP = 0;
        RX.OP_RX.R1X2 = 0;
        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);
                RAB = (char *) &RX.OP_RX.B2D2;
                swab(RAB, RAB, 2);
                STXT(4);
        } else if (!memcmp(TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND, "H'", 2))
        {
                RAB = ( (char*) TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND + 2,
"'");
                RR.OP_RR.OP = 0;
                RR.OP_RR.R1R2 = atoi(RAB);
                RAB = (char *) &RR.OP_RR.R1R2;
                STXT(2);
        }
        else if (!memcmp(TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND, "BL2'", 4))
        {

                RAB = strtok ((char*) TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND + 4, "'");
                int value = atoi(RAB);
                int len = strlen(RAB);
```

```
                value <<= (16 - len);
                RAB = (char *) &value;
                swab(RAB, RAB, 2);
                char buf[2] = { TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[2], '\x0' };
                int bytes = atoi(buf);
                RR.OP_RR.OP = 0;
                RR.OP_RR.R1R2 = 0;
                memcpy(RR.BUF_OP_RR, &value, 2);
                STXT(2);
        } else
                return (1);
        return (0);
}
```

2) Определение оператора DS на втором просмотре

```
int SDS()
{
        if ( TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == 'F')
        {
                RX.OP_RX.OP = 0;
                RX.OP_RX.R1X2 = 0;
                RX.OP_RX.B2D2 = 0;
                STXT(4);
        } else if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == 'B'
                && TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'L'
                && TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[2] == '2')
        {
                RR.OP_RR.OP = 0;
                RR.OP_RR.R1R2 = 0;
                STXT(2);
        } else if (TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[0] == '0'
                && TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND[1] == 'H' )
        {
                if (CHADR % 2)
                {
                        CHADR = (CHADR / 2 + 1) * 2;
                }
        } else
                return (1);
        return (0);
}
```

3) Определение операции RX на втором посмотре

```
 int SRX()
{
        char *METKA;
        char *METKA1;
        char *METKA2;
        char *PTR;
        int DELTA;
        int ZNSYM;
        int NBASRG;
        int J; int I;
        unsigned char R1X2;
        int B2D2;
        RX.OP_RX.OP = T_MOP[I3].CODOP;
        METKA1 = strtok ( (char*) TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAND, "," );
        METKA2 = strtok(NULL, " " );
        if (isalpha ( (int) *METKA1 ) || (int) *METKA1 == '@')
        {
                for (J = 0; J <= ITSYM; J++)
                {
                        METKA = strtok((char*) T_SYM[J].IMSYM, " ");
                        if (!strcmp(METKA, METKA1))
                        {
                                R1X2 = T_SYM[J].ZNSYM << 4;
                                goto SRX1;
                        }
                }
```

```
                return (2);
        } else
        {
                R1X2 = atoi(METKA1) << 4;
        }

        SRX1:

        if (isalpha ( (int) *METKA2 ) || (int) *METKA2 == '@')
        {
                for (J = 0; J <= ITSYM; J++)
                {
                        METKA = strtok((char*) T_SYM[J].IMSYM, " ");
                        if (!strcmp(METKA, METKA2))
                        {
                                NBASRG = 0;
                                DELTA = 0xfff - 1;
                                ZNSYM = T_SYM[J].ZNSYM;
                                for (I = 0; I < 15; I++)
                                {
                                        if (
                                        T_BASR[I].PRDOST == 'Y'       &&      ZNSYM - T_BASR[I].SMESH
>= 0 && ZNSYM - T_BASR[I].SMESH < DELTA )
                                        {
                                                NBASRG = I + 1;
                                                DELTA = ZNSYM - T_BASR[I].SMESH;

                                        }
                                }
                                if (NBASRG == 0 || DELTA > 0xfff)
                                        return (5);
                                else
                                {
                                        B2D2 = NBASRG << 12;
                                        B2D2 = B2D2 + DELTA;
                                        PTR = (char *) &B2D2;
                                        swab(PTR, PTR, 2);
                                        RX.OP_RX.B2D2 = B2D2;
                                }
                                goto SRX2;
                        }
                }
                return (2);
        } else if (isdigit(METKA2[0])) {
                NBASRG = 15;
                DELTA = atoi(&METKA2[0]);
                B2D2 = NBASRG << 12;
                B2D2 = B2D2 + DELTA;
                PTR = (char *) &B2D2;
                swab(PTR, PTR, 2);
                RX.OP_RX.B2D2 = B2D2;
        } else
        {
                return (4);
        }
        SRX2:
        printf("\noperc = %s ", TEK_ISX_KARTA.STRUCT_BUFCARD.OPERAC);
        printf("  BASE %d DELTA %d\n", NBASRG, DELTA);
        RX.OP_RX.R1X2 = R1X2;
        STXT(4);
        return (0);
}
```

Выводы

В рамках второго этапа курсовой работы по написанию компилятора с языка ассемблер были выполнены все поставленные задачи:

1) Расширена таблица машинных команд
2) Модифицированы обработчики команд DC и DS
3) Существующий компилятор доработан с учетом новых правил

В результате выполнения работы был получен объектный модуль. Проверить правильность его работы можно с помощью абсолютного загрузчика и эмулятора машины(третий этап курсовой работы). Замечания по коду сохраняются.