Минобрнауки России

Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

**Кафедра «Информационные и управляющие системы»**

**КУРСОВОЙ ПРОЕКТ**

Разработка учебной системы программирования

Вариант 7

**Абсолютный загрузчик, эмулятор и отладчик**

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 3](#_Toc356973846)

[План работы 3](#_Toc356973847)

[Модификация кода загрузчика 3](#_Toc356973848)

[Обработчики 5](#_Toc356973849)

[Выводы 6](#_Toc356973850)

# Задание

*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 второй части курсовой работы был получен объектный модуль. На данном этапе курсового проекта необходимо проверить работу модуля с помощью загрузчика, эмулятора и отладчика.

# План работы

Необходимо доработать загрузчик, дополнив его новой функциональностью. В новую функциональность входит поддержка новых команд (LH, STH, SRL, SLL, OR, NR).

# Модификация кода загрузчика

#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' , 2 , FRR} ,

{{'N','R',' ',' ',' '} , '\x14' , 2 , FRR} ,

{{'S','T','H',' ',' '} , '\x40' , 4 , FRX} ,

};

Также были установлены указатели на программные обработчики введённых машинных команд:

SKIP:

switch (T\_MOP[k].CODOP) //согласно коду команды,

{ //селектируемой сч.адреса

//выбрать подпрогр.интер-

case '\x05' : P\_BALR(); //претации семантики

break; //текущей команды

case '\x07' : { i = P\_BCR();

getch();

if (i == 1)

return 8;

}

break;

case '\x50' : P\_ST();

break;

case '\x58' : P\_L();

break;

case '\x5A' : P\_A();

break;

case '\x5B' : P\_S();

break;

case '\x48' : P\_LH();

break;

case '\x01' : P\_SRL();

break;

case '\x02' : P\_SLL();

break;

case '\x16' : P\_OR();

break;

case '\x14' : P\_NR();

break;

case '\x40' : P\_STH();

}

# Обработчики

1. Функция реализации семантики команды LH – загрузки полуслова в регистр

int P\_LH()

{

int sm;

/\* Вычисление абсолютного адреса операнда \*/

ADDR = VR[B] + VR[X] + D;

/\* Вычисление смещения операнда \*/

sm = (int)(ADDR - I);

/\* Загрузка операнда на регистр \*/

VR[R1] = OBLZ[BAS\_IND + CUR\_IND + sm] \* 0x100L + OBLZ[BAS\_IND + CUR\_IND + sm + 1];

return 0;

}

1. Функция реализации семантики команды SRL – сдвиг операнда вправо

int P\_SRL()

{

int sm = 3;

VR[R1] = VR[R1] >> sm;

return 0;

}

1. Функция реализации семантики команды SLL – сдвиг операнда влево

int P\_SLL()

{

int sm = 1;

VR[R1] = VR[R1] << sm;

return 0;

}

1. Функция реализации семантики команды OR – логическое ИЛИ регистров

int P\_OR()

{

VR[R1] = VR[R1] | VR[R2];

return 0;

}

1. Функция реализации семантики команды NR – логическое И регистров

int P\_NR()

{

VR[R1] = VR[R1] & VR[R2];

return 0;

}

1. Функция реализации семантики команды STH – выгрузка полуслова из регистра

int P\_STH()

{

int sm,i;

char bytes[2];

ADDR = VR[B] + VR[X] + D; /\*вычисление абс.адреса и \*/

sm = (int) (ADDR -I + 8); /\*смещения \*/

bytes[0] = ((VR[R1] % 0x10000L) - /\*преобразование содержим.\*/

((VR[R1]%0x10000L)%0x100))/0x100; /\*РОН, использованного в \*/

bytes[1] = (VR[R1] % 0x10000L) % 0x100; /\*качестве первого оп-да, \*/

/\*к виду, принятому в \*/

/\*ЕС ЭВМ \*/

for (i=0; i<2; i++) /\*запись преобразованого \*/

OBLZ[BAS\_IND + CUR\_IND + sm + i] = bytes[i]; /\*значения по адресу 2-г \*/

/\*операнда \*/

return 0;

}

# Выводы

В рамках третьего этапа курсовой работы по проверке работы полученного объектного модуля на втором этапе были выполнены все поставленные задачи:

1. Расширена таблица машинных команд
2. Модифицированы обработчики новых машинных команд
3. Существующий компилятор доработан с учетом новых правил
4. Осуществлена проверка работы модуля

В результате выполнения трех этапов курсовой работы были изучены компиляторы с языка высокого уровня на Ассемблер, с Ассемблера в объектное представление и абсолютный загрузчик. Их код был дополнен новой функциональностью, позволяющей решать новые задачи.