МИНОБРНАУКИ РОССИИ

ФГБОУ ВО «Ижевский государственный технический университет имени М. Т. Калашникова»

Кафедра «Программное обеспечение»

Отчет по лабораторной работе №2

## на тему «Проектирование Ассемблера для простых способов адресации операндов»

по дисциплине «Машинно-зависимые языки программирования»

## Вариант 5.

Выполнил:

студент группы Б04-191-3 Р. А. Гумметов

Принял:

к.т.н., профессор В. Г. Тарасов

Ижевск 2020

1. Цель работы

Изучение структур данных и алгоритмов, применяемых при проектировании транслятора с языка символического кодирования в машинный код.

1. Задание

Исходные данные содержат команды вида:

МНЕМА r,r

МНЕМА r,m

МНЕМА m,r

…

Здесь МНЕМА = [sub, cmp].

Операнд в памяти (m) может быть задан одним из следующих способов: [BX], [SI], [DI], [BX+SI], [BX+DI].

Требуется сформировать листинг для этого потока команд для случая 16-разрядных операндов.

1. Шаблоны машинных команд

add: cmp (рег/память с регистром):

|  |  |
| --- | --- |
| 001010dw | mod(2) reg(3) r/m(3) |

|  |  |
| --- | --- |
| 0011101w | mod(2) reg(3) r/m(3) |

cmp (регистр с рег/памятью):

|  |  |
| --- | --- |
| 0011100w | mod(2) reg(3) r/m(3) |

По условию задачи на вход не подаются команды, содержащие непосредственный операнд, поэтому другие шаблоны машинных команд не используются.

В случае операндов reg, regполе mod = 11, поле r/m используется для регистра. В случае reg, memили mem, regполе mod = 00 (так как DISP не обрабатывается) и поле r/mиспользуется для операнда в памяти.

1. Описание программы
2. Команды считываются из файла.
3. Каждая команда разделяется на строку с названием команды (sub, cmp) и на строки, содержащие ее операнды. Заранее предполагается, что между названием и операндами и между операндом и операндом всегда один пробел.
4. Далее проводится анализ операндов.
   1. В соответствии с именем операнда определяется местонахождение данного операнда – в памяти или в регистре.
   2. Для операнда в памяти определяется способ его задания: [BX], [SI], [DI], [BX+SI] или [BX+DI]
   3. В случае с операндом в регистре необходимо запомнить номер регистра в таблице регистров regt и определить значение бита wв соответствии с данным номером.
5. В соответствии с проведенным анализом синтезируется машинная команда.
   1. mod= 11, если операнды в регистрах; mod = 00 – в памяти.
   2. Поле regопределяется по номеру регистра в таблице.
   3. Поле r/mопределяется либо по номеру второго регистра, либо, если операнд в памяти, по номеру алгоритма, с помощью которого вычисляется способ задания операнда в памяти.
6. Машинная команда выводится на экран в шестнадцатеричном виде.
7. Текст программы

#include<iostream>

#include<string>

#include<vector>

#include<algorithm>

#include<fstream>

#include<iomanip>

#defineMEM\_T 5

#defineOPCODE 2

usingnamespacestd;

boolreg[2], mem[2], bx[2], bx\_si[2], bx\_di[2], si[2], di[2], w, d;

intregs[2];

voidclear()

{

for (size\_ti = 0; i<2; i++)

{

reg[i] = false; mem[i] = false; bx[i] = false; bx\_si[i] = false;

bx\_di[i] = false; si[i] = false; di[i] = false;

}

w = false;

d = false;

}

chart[4] = "000";

void i2b(charx)

{

char m = 1;

for (size\_ti = 0; i< 3; i++)

{

if (x& m) t[2 - i] = '1';

elset[2 - i] = '0';

m = m << 1;

}

}

charregt[16][3] = { "al", "cl", "dl", "bl", "ah", "ch", "dh", "bh", "ax", "cx", "dx", "bx", "sp", "bp", "si", "di" };

structopcode\_t

{

string name;

string code;

};

opcode\_t op[OPCODE] =

{

{"sub", "001010"},

{"cmp", "001110"}

};

structmem\_t

{

boolbx;

boolbx\_si;

boolbx\_di;

boolsi;

bool di;

stringr\_m;

};

mem\_tmemt[MEM\_T] =

{

{true, false, false, false, false, "111"},

{true, true, false, false, false, "000"},

{true, false, true, false, false, "001"},

{false, false, false, true, false, "100"},

{false, false, false, false, true, "101"}

};

voidboth\_reg(string&bin)

{

bin+="11";

i2b(regs[0]);

bin+= t;

i2b(regs[1]);

bin+= t;

}

voidreg\_mem(string&bin)

{

bin+="00";

size\_t m, r;

if (d)

{

m = 1;

r = 0;

}

else

{

m = 0;

r = 1;

}

i2b(regs[r]);

bin+= t;

for (size\_ti = 0; i<MEM\_T; i++)

{

if (memt[i].bx == bx[m] &&memt[i].bx\_di == bx\_di[m] &&memt[i].bx\_si == bx\_si[m]

&&memt[i].di == di[m] &&memt[i].si == si[m])

{

bin+=memt[i].r\_m;

}

}

}

voidoperand(intn, string&operand)

{

n--;

if (operand.find('[') != -1)

{

mem[n] = 1;

if (operand.find("bx") != -1)

{

bx[n] = 1;

if (operand.find("bx + si") != -1)

bx\_si[n] = 1;

elseif (operand.find("bx + di") != -1)

bx\_di[n] = 1;

}

elseif (operand.find("si") != -1)

si[n] = 1;

elseif (operand.find("di") != -1)

di[n] = 1;

}

else

{

reg[n] = 1;

for (size\_ti = 0; i< 16; i++)

if (operand==regt[i])

regs[n] = i;

w = (regs[n] >> 3);

}

}

voidget\_command(string&bin, string&name)

{

for (size\_ti = 0; i<OPCODE; i++)

{

if (op[i].name ==name)

bin= op[i].code;

}

if (reg[0] &&reg[1] || reg[0] && mem[1])

d = true;

else

d = false;

bin+=to\_string((int)d);

bin+=to\_string((int)w);

if (reg[0] &&reg[1])

both\_reg(bin);

elseif (reg[0] && mem[1] || reg[1] && mem[0])

reg\_mem(bin);

}

int main()

{

setlocale(LC\_ALL, "Russian");

ifstreamrf("c.txt");

stringcmd;

stringbin\_str;

longintlongint;

cout<<setbase(16);

while (getline(rf, cmd))

{

string name;

stringoperand\_name;

int j = 0, start\_operand;

for (; j <cmd.size(); j++)

{

if (cmd[j] == ' ')

{

name =cmd.substr(0, j);

j++;

start\_operand = j;

break;

}

}

clear();

for (; j <cmd.size(); j++)

{

if (cmd[j] == ',')

{

operand\_name=cmd.substr(start\_operand, j - start\_operand);

j+=2;

start\_operand = j;

operand(1, operand\_name);

}

elseif (j == cmd.size() - 1)

{

operand\_name=cmd.substr(start\_operand, j - start\_operand + 1);

operand(2, operand\_name);

}

}

cout<<"Исходная команда: "<<cmd<<endl;

get\_command(bin\_str, name);

longint = 0;

intlen = bin\_str.size();

for (inti = 0; i<len; i++)

longint += (bin\_str[len - i - 1] - 48) \* pow(2, i);

cout<<"Машиннаякоманда: "<<longint<<endl<<endl;

}

rf.close();

return 0;

}

1. Результаты работы программы

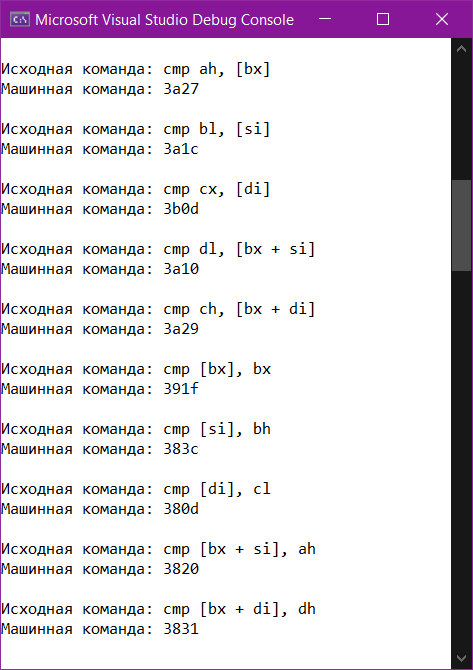
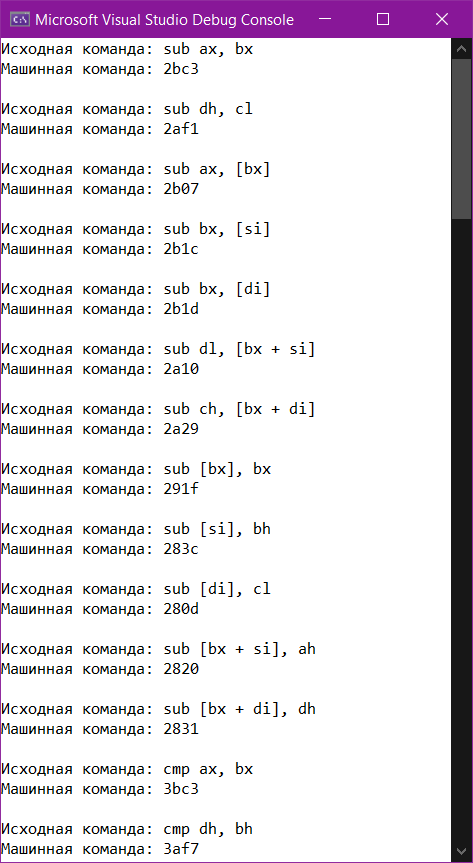


Рис. 2. Результаты работы программы (2)

Рис. 1. Результаты работы программы.

Для сравнения на рис. 3 приведен листинг стандартного ассемблера (опущено несколько команд, в которых регистры заполняются начальными значениями).

TurboAssembler Version 3.2 06/19/20 13:36:28 Page 1

a.asm

1 0000 csegsegment

2 assume cs:cseg

3 0000 start:

...

8

9 000A 2B C3 sub ax, bx

10 000C 2A F1 sub dh, cl

11

12 000E 2B 07 sub ax, [bx]

13 0010 2B 1C sub bx, [si]

14 0012 2B 1D sub bx, [di]

15 0014 2A 10 sub dl, [bx + si]

16 0016 2A 29 sub ch, [bx + di]

17

18 0018 29 1F sub [bx], bx

19 001A 28 3C sub [si], bh

20 001C 28 0D sub [di], cl

21 001E 28 20 sub [bx + si], ah

22 0020 28 31 sub [bx + di], dh

23

24 0022 3B C3 cmp ax, bx

25 0024 3A F7 cmp dh, bh

26

27 0026 3A 27 cmp ah, [bx]

28 0028 3A 1C cmpbl, [si]

29 002A 3B 0D cmp cx, [di]

30 002C 3A 10 cmp dl, [bx + si]

31 002E 3A 29 cmpch, [bx + di]

32

33 0030 39 1F cmp [bx], bx

34 0032 38 3C cmp [si], bh

35 0034 38 0D cmp [di], cl

36 0036 38 20 cmp [bx + si], ah

37 0038 38 31 cmp [bx + di], dh

38

39 003A B8 4C00 mov ax, 04C00h

40 003D CD 21 int 21h

41 003F cseg ends

42 end start

Рис. 3.Листинг кода в tasm

1. Вывод

В данной лабораторной работе были изучены основные структуры данных и алгоритмы, применяемые при проектировании транслятора с языка символического кодирования в машинный код. На вход подавался набор команд на языке ассемблера, ограниченный определенными условиями: команды sub, cmp с операндами, находящимися в регистрах и в оперативной памяти.