**Машинно-зависимая и машинно-независимая оптимизация кода ассемблера**

### 1. Постановка задачи

Лабораторная работа №3: Машинно-зависимая и машинно-независимая оптимизация кода ассемблера.

На языке PASCAL написана следующая программа, предназначенная для определения нечетных чисел находящихся в массиве чисел:

var

res, i: integer;

begin

res:=0;

for i:=1 to 10 do

if i mod 2 <> 0 then

begin

res:=i;

writeln (res);

end;

end.

**2. Оптимизация программы**

Листинг программы, полученной с помощью программы ALTER, для последующей оптимизации (подчеркнуты строки, измененные в процессе оптимизации):

Turbo Assembler Version 4.1 12/12/08 00:18:57 Page 1

11. ASM

1 0000 .Model Small

2 0000 .Stack 100h

3 0000 .Data

4 0000 FF\*(??) \_\_bufrw db 255 dup (?)

5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;\_res\_\_\_\_\_ dw\_\_\_\_\_?

6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;\_i\_\_\_\_\_ dw\_\_\_\_\_?

Удалить строки 5 и 6

7 00FF .Code

8 0000 Main proc

9 0000 B8 0000s mov ax,@data

10 0003 8E D8 mov ds, ax

11 ; Присвоить переменной \_res

12\_\_\_\_\_\_\_\_\_\_; **mov \_res, 0**\_\_\_\_\_; Занести в\_\_\_\_\_переменную \_res\_\_\_\_\_0

Строку 12 заменить на **mov ax, 0**

13 0005 B8 0000

14 ; Начало цикла FOR №1

15 ; Присвоить переменной \_i

16\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; **mov \_i, 1**\_\_\_\_\_; Занести в\_\_\_\_\_переменную \_i 1

17\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; **mov bx,\_i**; Записать в BX \_i

Строки 16, 17 заменить на **mov bx, 1**

18 0008 BB 0001

19 000B B9 000A mov cx, 10 ; Занести в cx 10

20 000E EB 01 jmp $+3 ; Перейти на 3 байта вперед

21 0010 for\_b\_1:

22 0010 43 inc bx ; Инкремент BX

23\_\_\_\_\_\_\_\_\_\_; **push bx**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; Записать BX в стек

24\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; **push cx**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; Записать CX в стек

Строки 23, 24 заменить на **push bx cx**

25 0011 53 51

26\_\_\_\_\_; **mov \_i, bx**\_\_\_\_\_\_\_\_\_\_; Записать в переменную \_i BX

Удалить строку 26

27 ; Обработка условия №1

28 ; Подготовка левой части условия

29 ; mov ax,\_i ; Занести в AX переменную \_i

30 0013 8B C3 mov ax, bx

31 0015 53 push bx

32 0016 BB 0002 mov bx, 2 ; Занести в BX 2

33 0019 33 D2 xor dx, dx ; Обнуляем DX

34 001B F7 F3 div bx ; Деление без знака

35 001D 92 xchg ax, dx ; Поменять значение AX и BX

36 001E 8B D8 mov bx, ax ; Записать в bx ax

37 ; Подготовка правой части условия

38 0020 BA 0000 mov dx, 0 ; Занести в dx 0

39 0023 3B DA cmp bx, dx ; Сравнить BX c DX

40 0025 5B pop bx

41 0026 74 0F je else\_1

42 ; Присвоить переменной \_res

43\_\_\_\_\_\_\_\_\_\_; mov ax,\_i\_\_\_\_\_\_\_\_\_\_; Занести в\_\_\_\_\_AX переменную \_i

Заменить на строку **mov ax, bx**

44 0028 8B C3

45\_\_\_\_\_; **mov \_res, ax**\_\_\_\_\_; Занести в\_\_\_\_\_переменную \_res\_\_\_\_\_AX

Удалить строку 45

46 ; Обработка процедуры WRITE

47\_\_\_\_\_\_\_\_\_\_; **mov ax,\_res**\_\_\_\_\_; Занести в\_\_\_\_\_AX переменную \_res

Удалить строку 45

48 002A E8 0014 call sl\_itoa ; Вызвать функцию преобразования

49 002D B4 02 mov ah, 2 ; Загрузить в AH номер функции

50 002F B2 0D mov dl, 13 ; Выводимый символ

51 0031 CD 21 int 21h ; Вызвать прерывание DOS

52 0033 B2 0A mov dl, 10 ; Выводимый символ

53 0035 CD 21 int 21h ; Вызвать прерывание DOS

54 0037 else\_1:

55 ; Переход если не выполнилось условие №1

56\_\_\_\_\_\_\_\_\_\_; **pop cx\_\_\_\_\_**\_\_\_\_\_; Вернуть из стека в CX

57\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; **pop bx**\_\_\_\_\_\_\_\_\_\_; Вернуть из стека в BX

Заменить на строку **pop cx bx**

Turbo Assembler Version 4.1 12/12/08 00:18:57 Page 2

11.ASM

58 0037 59 5B

59 0039 3B D9 cmp bx, cx ; Сравнить BX и CX

60 003B 75 D3 jne for\_b\_1 ; Переход если не равно

61 ; Конец цикла FOR №1

62 003D B4 4C mov ah, 4ch

63 003F CD 21 int 21h

64 0041 endp

65 0041 sl\_itoa proc

66 0041 50 53 51 52 push ax bx cx dx

67 0045 3D 0000 cmp ax, 0

68 0048 7D 0A jge Doit

69 004A 50 push ax

70 004B B2 2D mov dl, '-'

71 004D B4 02 mov ah, 2

72 004F CD 21 int 21h

73 0051 58 pop ax

74 0052 F7 D8 neg ax

75 0054 E8 0011 DoIt: call puti2

76 0057 5A 59 5B 58 pop dx cx bx ax

77 005B C3 ret

78 005C sl\_itoa endp

79

80 005C sl\_utoa proc

81 005C 50 53 51 52 push ax bx cx dx

82 0060 E8 0005 call PutI2

83 0063 5A 59 5B 58 pop dx cx bx ax

84 0067 C3 ret

85 0068 sl\_utoa endp

86

87 0068 Puti2 proc

88 0068 BB 000A mov bx, 10

89 006B 33 D2 xor dx, dx

90 006D F7 F3 div bx

91 006F 0B C0 or ax, ax

92 0071 74 05 jz Done

93 0073 52 push dx

94 0074 E8 FFF1 call Puti2

95 0077 5A pop dx

96 0078 Done:

97 0078 80 CA 30 or dl, '0'

98 007B B4 02 mov ah, 2

99 007D CD 21 int 21h

100 007F C3 ret

101 0080 PutI2 endp

102

103 end Main

Turbo Assembler Version 4.1 12/12/08 00:18:57 Page 3

Symbol Table

Symbol Name Type Value

?? DATE Text «12/12/08»

?? FILENAME Text «11 »

?? TIME Text «00:18:56»

?? VERSION Number 040A

@32BIT Text 0

@CODE Text \_TEXT

@CODESIZE Text 0

@CPU Text 0101H

@CURSEG Text \_TEXT

@DATA Text DGROUP

@DATASIZE Text 0

@FILENAME Text 11

@INTERFACE Text 000H

@MODEL Text 2

@STACK Text DGROUP

@WORDSIZE Text 2

DOIT Near \_TEXT:0054

DONE Near \_TEXT:0078

ELSE\_1 Near \_TEXT:0037

FOR\_B\_1 Near \_TEXT:0010

MAIN Near \_TEXT:0000

PUTI2 Near \_TEXT:0068

SL\_ITOA Near \_TEXT:0041

SL\_UTOA Near \_TEXT:005C

\_\_BUFRW Byte DGROUP:0000

Groups & Segments Bit Size Align Combine Class

DGROUP Group

STACK 16 0100 Para Stack STACK

\_DATA 16 00FF Word Public DATA

\_TEXT 16 0080 Word Public CODE

**3. Машинно-зависимая оптимизация кода ассемблера**

Машинно-зависимая оптимизация предполагает:

1. Использование регистров процессора вместо ячеек оперативной памяти.
2. Использование непосредственно заданных операндов.
3. Использование косвенной адресации, когда операнд хранит адрес операнда.
4. Ограничение использования стека.

программа оптимизация ассемблер код

**4. Машинно-независимая оптимизация кода ассемблера**

Машинно-независимая оптимизация предполагает:

1. Одним из важнейших источников оптимизации кода является удаление общих подвыражений, т.е. подвыражений, которые встречаются в нескольких местах программы и вычисляют одно и то же выражение.

2. Другим источником оптимизации кода является удаление инвариантов цикла. Так называются подвыражения внутри цикла, результирующие значения которых не изменяются внутри цикла при переходе от одной итерации к другой. Поскольку для большинства программ основное время работы приходится на выполнение циклов, экономия времени от подобной оптимизации может быть весьма существенной.

3. Еще один источник оптимизации кода состоит в замене менее эффективных операций на более эффективные.

**Выводы**

До оптимизации исполняемый файл составлял 660 байт, а после оптимизации 640 байт.

В ходе оптимизации были произведены следующие действия:

– вместо переменных i, res используются регистры ax, bx, что является машинно-зависимой оптимизацией.

– удаление общих подвыражений, были объединены такие строки, как

pop сx

pop bx

в одну – pop сx bx;

push cx

push bx

в одну – push bx cx;

После проведенной оптимизации программа полностью сохранила свой смысл.