

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение  
высшего образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО  
ОБРАЗОВАНИЯ  
КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Оценка

Преподаватель

ст. преподаватель

(должность, ученое звание)

Д.В. Куртяник

(И.О. Фамилия)

Отчет о лабораторной работе №4

Разработка встраиваемой программы

по дисциплине «Программирование на языках Ассемблера»

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № Z2442

(номер группы)

Д.О. Максимов

(И.О. Фамилия)

Санкт-Петербург 2024

# 1 Цель работы

Изучение связи ассемблера с языками высокого уровня, получение базовых навыков работы с ассемблерными вставками. Вариант 16.  
Вычислить сумму ряда  $1 + 2 + 4 + 7 + \dots + n$  при заданном  $n$ .

## 2 Описание алгоритма

Данная последовательность называется центральными многоугольными числами, где каждый член последовательности может быть найден по формуле.

$$a(n) = \frac{n*(n+1)}{2} + 1$$

Соответственно для поиска суммы нужно организовать цикл в котором будет генерироваться  $n$ -ый член прогрессии и добавляться в итоговую сумму. Цикл реализован инструкций loop. В качестве счетчика используется регистр сх. Код генерации  $n$ -ого члена прогрессии вынесен в отдельную процедуру или макрос.

### 3 Исходный код

---

```
1 #include "stdio.h"
2 #include "stdint.h"
3
4
5 int main() {
6     uint64_t n = 4, sum = 0, index = 0;
7     scanf("%ld", &n);
8     printf("%ld\n", n);
9     asm (
10         "movq %1, %%rcx ;"
11         "mainloop :"
12         "    movq %2, %%rax ;"
13         "    incq %%rax ;"
14         "    mulq %2; "
15         "    movq $2, %%rbx ;"
16         "    divq %%rbx, %%rax ;"
17         "    incq %%rax ;"
18         "    movq %3, %%rbx ;"
19         "    addq %%rax, %%rbx ;"
20         "    movq %%rbx, %3; "
21         "    movq %2, %%rax ;"
22         "    incq %%rax ;"
23         "    movq %%rax, %2; "
24         "    loopq mainloop ;"
25         "    movq %3, %%rax ;"
26         "    movq %%rax, %0 ;"
27
28     : "=r" (sum)
29     : "r" (n), "r" (index), "r" (sum)
30     : "rax", "rbx", "rcx", "rdx"
31 );
32     printf("sum = %ld\n", sum);
33     return 0;
34 }
```

---

Листинг 1 — Исходный код

## 4 Анализ работы

### 4.1 Версия с процедурой

В начале программы по адресу CS:000B задается количество выполнений n. Далее в CS:0013 выполняется переход в тело процедуры CS:002D. На рисунке 2 генерируется n-ый элемент и возврат к CS:0016. Далее в CS:0021 инструкция LOOP сначала из регистра CX вычитает единицу и затем его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке адресу CS:000F. Иначе переход не выполняется и управление передается команде, которая следует сразу после команды LOOP. В конце сумма сохраняется в перемену по адресу DS:000E.

The screenshot shows a debugger interface with the following details:

- Registers:** ax = 0000, bx = 0000, cx = 0000, dx = 0000, si = 0000, di = 0000, bp = 0000, sp = 0100, ds = 1817, es = 1817, ss = 182C, cs = 1827, ip = 0000.
- Stack:** ss:0102 0FC3, ss:0100 5859, ss:00FE 0000, ss:00FC 5B5D, ss:00FA 5E5F.
- Memory Dump:** ds:0000 CD 20 FB 9F 00 9A F0 FE = JA   , ds:0008 1D F0 32 0B 06 15 0F 07 + 2    \*, ds:0010 82 12 56 01 2A 07 65 12 B U      , ds:0018 01 01 01 00 02 04 FF FF       , ds:0020 FF FF FF FF FF FF FF FF.
- Assembly View:** The assembly code starts at address CS:000B with the instruction MOVS AX, 182B. It then moves values from memory into registers (DS, AX, AX, DX, CX), pushes a word to memory, and calls CS:002D. The next instruction is ADD DX, AX, followed by MOVS AX, [000C], which is highlighted in red. This is followed by ADD AX, 0001, MOVS [000C], AX, LOOP 000F, MOVS [000E], DX, MOVS AL, 0001, and MOVS AH, 4C.
- Bottom Bar:** F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Рисунок 1 — Начало программы с процедурой

The screenshot shows a debugger interface with the title bar "READY". The assembly code window displays the following sequence of instructions:

```

cs:002D 55          push    bp
cs:002E 8BEC        mov     bp,sp
cs:0030 53          push    bx
cs:0031 52          push    dx
cs:0032 8B4604      mov     ax,[bp+04]
cs:0035 83C001      add    ax,0001
cs:0038 F76604      mul    word ptr [bp+04]
cs:003B BB0200      mov     bx,0002
cs:003E F7F3        div    bx
cs:0040 83C001      add    ax,0001
cs:0043 5A          pop    dx
cs:0044 5B          pop    bx
cs:0045 5D          pop    bp
cs:0046 C20200      ret    0002
cs:0049 0011        add    [bx+di],dl

```

Registers and memory dump areas are visible on the right side of the interface.

Рисунок 2 — Тело процедуры

The screenshot shows a debugger interface with the title bar "READY". The assembly code window displays the following sequence of instructions:

```

cs:0005 B80000      mov     ax,0000
cs:0008 BA0000      mov     dx,0000
cs:000B 8B0E0A00      mov    cx,[000A]
cs:000F FF360C00      push   word ptr [000C]
cs:0013 E81700      call   002D
cs:0016 83D0          add    dx,ax
cs:0018 A10C00      mov    ax,[000C]
cs:001B 83C001      add    ax,0001
cs:001E A30C00      mov    [000C],ax
cs:0021 E2EC          loop   000F
cs:0023 89160E00      mov    [000E],dx
cs:0027 B001          mov    al,01
cs:0029 B44C          mov    ah,4C
cs:002B CD21          int    21
cs:002D 55          push   bp

```

Registers and memory dump areas are visible on the right side of the interface.

Рисунок 3 — Конец программы

## 4.2 Результаты выполнения

The screenshot shows the Turbo Debugger interface. The assembly code window displays the following instructions:

```

cs:0016 03D0      add    dx,ax
cs:0018 A10C00     mov    ax,[000C]
cs:001B 83C001     add    ax,0001
cs:001E A30C00     mov    [000C],ax
cs:0021 E2EC      loop   000F
cs:0023 89160E00   mov    [000E],dx
cs:0027 B001      mov    al,01
cs:0029 B44C      mov    ah,4C
cs:002B CD21      int    21
cs:002D 55        push   bp
cs:002E 8BEC      mov    bp,sp
cs:0030 53        push   bx
cs:0031 52        push   dx
cs:0032 8B4604   mov    ax,[bp+04]
cs:0035 83C001   add    ax,0001

```

The registers window shows the following values:

ax	0001	c=0
bx	0000	z=0
cx	0000	s=0
dx	000E	o=0
si	0000	p=0
di	0000	a=0
bp	0000	i=1
sp	0100	d=0
ds	182B	
es	1817	
ss	182C	
cs	1827	
ip	0029	

The memory dump window shows the following bytes at address ds:000E:

```

ds:000E 0E 00 01 00 0A BE E4 39 Л © Сяф9
ds:0016 33 FF 8B 04 3B 46 16 74 3 Л♦;F-t
ds:001E 09 47 83 C6 0C 83 FF 10 оГГ;ФГ ►
ds:0026 7C F0 83 FF 10 75 03 E9 ;ЕГ ►иш
ds:002E D6 00 57 E8 FC 03 59 3B п WmНшY;

```

The status bar at the bottom shows: F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu.

Рисунок 4 — Результат при  $n = 4$

The screenshot shows the Turbo Debugger interface. The assembly code window displays the following instructions:

```

cs:0023 89160E00   mov    [000E],dx
cs:0027 B001       mov    al,01
cs:0029 B44C       mov    ah,4C
cs:002B CD21       int    21
cs:002D 55         push   bp
cs:002E 8BEC       mov    bp,sp
cs:0030 53         push   bx
cs:0031 52         push   dx
cs:0032 8B4604   mov    ax,[bp+04]
cs:0035 83C001   add    ax,0001
cs:0038 F76604   mul    word ptr [bp+04]
cs:003B BB0200   mov    bx,0002
cs:003E F7F3      div    bx

```

The registers window shows the following values:

ax	0011	c=0
bx	0000	z=0
cx	0000	s=0
dx	0341	o=0
si	0000	p=1
di	0000	a=0
bp	0000	i=1
sp	0100	d=0
ds	182B	
es	1817	
ss	182C	
cs	1827	
ip	0027	

The memory dump window shows the following bytes at address ds:000E:

```

ds:000E 0341 0001 BE0A 39E4
ds:0016 FF33 048B 463B 7416
ds:001E 4709 C683 830C 10FF
ds:0026 F07C FF83 7510 E903

```

The status bar at the bottom shows: F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu.

Рисунок 5 — Результат при  $n = 17$

### 4.3 Версия с макросом

В начале программы по адресу CS:000B задается количество выполнений  $n$ . Далее с CS:0011 по CS:0020 генерируется  $n$ -ный элемент. Далее в CS:0030 инструкция LOOP сначала из регистра CX вычитает единицу и затем его значение сравнивается с нулём. Если регистр не равен нулю, то

выполняется переход к указанной метке адресу CS:000F. Иначе переход не выполняется и управление передается команде, которая следует сразу после команды LOOP. В конце сумма сохраняется в перемену по адресу DS:0010.

The screenshot shows the CPU Pro assembly debugger interface. The assembly code in the main window is:

```

E File Edit View Run Breakpoints Data Options Window Help READY
[CPU P?ntium Pro]
cs:0003>8ED8    mov ds,ax
cs:0005 B80000    mov ax,0000
cs:0008 BA0000    mov dx,0000
cs:000B 8B0E0C00  mov cx,[000C]
cs:000F 53        push bx
cs:0010 52        push dx
cs:0011 A10E00    mov ax,[000E]
cs:0014 83C001    add ax,0001
cs:0017 F7260E00  mul word ptr [000E]
cs:001B BB0200    mov bx,0002
cs:001E F7F3      div bx
cs:0020 83C001    add ax,0001
cs:0023 5A        pop dx
cs:0024 5B        pop bx
cs:0025 03D0    add dx,ax

```

The registers window on the right shows:

ax	182A	c=0
bx	0000	z=0
cx	0000	s=0
dx	0000	o=0
si	0000	p=0
di	0000	a=0
bp	0000	i=1
sp	0100	d=0
ds	1817	
es	1817	
ss	182C	
cs	1827	
ip	0003	

The stack dump window at the bottom shows memory starting at address ds:0000:

```

ds:0000 CD 20 FB 9F 00 9A F0 FE = JЯ ЪЁ■
ds:0008 1D F0 32 0B 06 15 0F 07 +È2δ+§*.
ds:0010 82 12 56 01 2A 07 65 12 B†U@*.*e‡
ds:0018 01 01 01 00 02 04 FF FF @@@ @@+
ds:0020 FF FF FF FF FF FF FF FF FF

```

At the bottom is a keyboard status bar with keys F1-F10 labeled: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

Рисунок 6 — Начало программы с процедурой

The screenshot shows the CPU Pro assembly debugger interface. The assembly code in the main window is:

```

E File Edit View Run Breakpoints Data Options Window Help READY
[CPU P?ntium Pro]
cs:001E F7F3      div bx
cs:0020 83C001    add ax,0001
cs:0023 5A        pop dx
cs:0024 5B        pop bx
cs:0025 03D0    add dx,ax
cs:0027 A10E00    mov ax,[000E]
cs:002A 83C001    add ax,0001
cs:002D A30E00    mov [000E],ax
cs:0030 E2DD      loop 000F
cs:0032 89161000  mov [0010],dx
cs:0036 B001      mov al,01
cs:0038>B44C     mov ah,4C
cs:003A CD21      int 21
cs:003C 0F000F    str [bx]
cs:003F 003F      add [bx],bh

```

The registers window on the right shows:

ax	0001	c=0
bx	0000	z=0
cx	0000	s=0
dx	023F	o=0
si	0000	p=1
di	0000	a=0
bp	0000	i=1
sp	0100	d=0
ds	182A	
es	1817	
ss	182C	
cs	1827	
ip	0038	

The stack dump window at the bottom shows memory starting at address ds:0010:

```

ds:0010 023F 5A01 5D5B 02C2
ds:0018 0000 0011 0000 0000
ds:0020 EC83 BE0A 39E4 FF33
ds:0028 048B 463B 7416 4709
ds:0030 C683 830C 10FF F07C

```

At the bottom is a keyboard status bar with keys F1-F10 labeled: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

Рисунок 7 — Конец программы

## 4.4 Результаты выполнения

The screenshot shows the Turbo Debugger for DOS interface. The assembly code window displays the following instructions:

```

cs:0030 E2DD      loop    000F
cs:0032 89161000  mov     [0010],dx
cs:0036 B001      mov     al,01
cs:0038▶B44C    mov     ah,4C
cs:003A CD21      int     21
cs:003C 0400      add     al,00
cs:003E 0400      add     al,00
cs:0040 0000      add     [bx+sil,al]
cs:0042 58        pop    ax
cs:0043 C3        ret
cs:0044 50        push   ax
cs:0045 51        push   cx
cs:0046 52        push   dx
cs:0047 53        push   bx
cs:0048 54        push   sp

```

The registers window on the right shows the following values:

ax	0001	c = 0
bx	0000	z = 0
cx	0000	s = 0
dx	000E	o = 0
si	0000	p = 0
di	0000	a = 0
bp	0000	i = 1
sp	0100	d = 0
ds	182A	
es	1817	
ss	182C	
cs	1827	
ip	0038	

The stack dump window at the bottom shows the following memory dump:

```

ds:0006 01B0 4CB4 21CD 0004
ds:000E 0004 000E C358 5150
ds:0016 5352 5554 5756 061E
ds:001E EC8B EC83 BE0A 39E4
ds:0026 FF33 048B 463B 7416

```

At the bottom, the status bar shows keyboard shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

Рисунок 8 — Результат при n = 4

The screenshot shows the Turbo Debugger for DOS interface. The assembly code window displays the following instructions:

```

cs:0014 83C001    add    ax,0001
cs:0017 F7260E00  mul    word ptr [000E]
cs:001B BB0200    mov    bx,0002
cs:001E F7F3      div    bx
cs:0020 83C001    add    ax,0001
cs:0023 5A        pop    dx
cs:0024 5B        pop    bx
cs:0025 03D0      add    dx,ax
cs:0027 A10E00    mov    ax,[000E]
cs:002A 83C001    add    ax,0001
cs:002D A30E00    mov    [000E],ax
cs:0030 E2DD      loop   000F
cs:0032 89161000  mov    [0010],dx
cs:0036▶B001    mov    al,01
cs:0038 B44C      mov    ah,4C

```

The registers window on the right shows the following values:

ax	000F	c = 0
bx	0000	z = 0
cx	0000	s = 0
dx	023F	o = 0
si	0000	p = 1
di	0000	a = 0
bp	0000	i = 1
sp	0100	d = 0
ds	182A	
es	1817	
ss	182C	
cs	1827	
ip	0036	

The stack dump window at the bottom shows the following memory dump:

```

ds:0006 01B0 4CB4 21CD 000F
ds:000E 000F 023F C358 5150
ds:0016 5352 5554 5756 061E
ds:001E EC8B EC83 BE0A 39E4
ds:0026 FF33 048B 463B 7416

```

At the bottom, the status bar shows keyboard shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

Рисунок 9 — Результат при n = 15