ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе № 7

по дисциплине «Организация ЭВМ и систем»

Тема: Преобразование целых чисел. Использование процедур в Ассемблере.

Студент гр. 1303 _____ Ягодаров М.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Составить программу для преобразования чисел из одной заданной системы счисления в другую.

Задание.

Разработать на языке Ассемблер две процедуры: одна прямого и другая обратного преобразования целого числа, заданного в регистрах DX:AX. Преобразование провести без учёта знака. Система счисления для символьного изображения числа — восьмеричная. Связь данных между основной программой и подпрограммами осуществляется в первом случае с помощью РОН и общих данных, а во втором случае через РОН.

Выполнение работы.

Для удобства число в программу следует подавать в переменную `input_data`, лежащую в сегменте данных. После данное число передаётся в регистры dx:ax, а содержимое этих регистров с помощью процедуры `display_dx_ax` отображается в консольном выводе.

Поскольку нам необходимо выводить число без знака, исходные данные проверяются на отрицательное значение (старший бит равен единице): если число отрицательное, то оно с помощью процедуры `invert_number` преобразуется в положительное.

Далее, исходное число следует перевести в строчное представление, причём с использование восьмеричной системы счисления, — это производится с помощью процедуры `make_str`. В ней, начиная с конца строки, считываются по три бита (чтобы цифра не превышала восьми), к которым после прибавляется символ нуля, после чего полученное значение записывается в строку `number_str`. Стык dx:ax обрабатывается отдельно.

Затем полученная строка `number_str` с помощью системного вызова `syscall` под номером 1 выводится в консоль.

После необходимо обратно записать число в полученной строке в

регистры dx:ax — это вполняется с помощью процедуры `get_number`. В ней посимвольно считываются символы строки, из которых вычитается символ нуля; далее, символ записывается в регистр, после чего значение в регистре сдвигается влево на 3 (за исключением стыка dx:ax), а также последнего символа, записываемого в ax.

В конце концов, содержимое регистров dx:ax вновь выводится на экран, а затем программа с помощью системного вызова под номером 60 завершается.

Выводы

Разработана программа, преобразующая число из регистров dx:ax в восьмеричное число, представленное в символьном строчном отображении и обратно.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММ

Название файла: main.s

```
.section .rodata
 newline_str:
   .string "\n"

.section .data
 input_number:
   .long -1350000000

#; Uninitialized data
.bss
 number_str:
   .space 11  #; first symbol is 8 if num < 0
   .set number_str_len, 11
 dx_ax_str:
   .space 40
   .set dx_ax_str_len, 40

.section .text

#; Input:
#; dx:ax -- number
#; Output:
#; number_str -- number to oct string
make_str:
 push cx
 push rdi
 std

 mov  rdi, offset number_str
 add  rdi, 10

#; process ax
 mov  cl, 5
ax_loop:
 push ax

 and  ax, 0x7
 add  ax, '0'
 stosb

 pop  ax
 shr  ax, 3

 loop ax_loop
```

```
#; between ax and dx
 push dx

 and  dx, 0×3
 shl  dx, 1
 add  ax, dx
 add  ax, '0'
 stosb

 pop  dx
 mov  ax, dx
 shr  ax, 2

#; process dx
 mov  cl, 5
dx_loop:
 push ax

 and  ax, 0×7
 add  ax, '0'
 stosb

 pop  ax
 shr  ax, 3

 loop dx_loop

 cld
 pop  rdi
 pop  cx

 ret

#; Input:
#; rsi -- input str
#; Output:
#; dx:ax -- number
get_number:
 push rbx
 push rcx
 xor  rdx, rdx
 xor  rbx, rbx
 xor  rcx, rcx

 mov  cx, 4
proceed_digit_dx:
 lodsb
 sub  ax, '0'
 add  dx, ax
 shl  dx, 3
 loop proceed_digit_dx
```

```
 lodsb
 sub ax, '0'
 add dx, ax
 shl dx, 2

 lodsb
 sub ax, '0'
 mov bx, ax
 shr ax, 1
 add dx, ax

 mov cx, 5
proceed_digit_ax:
 lodsb
 sub ax, '0'
 shl bx, 3
 add bx, ax
 loop proceed_digit_ax

 mov rax, rbx

 pop rcx
 pop rbx
 ret

invert_number:
 test ax, ax
 jz  ax_null
 dec ax
 jmp neg_number

ax_null:
 dec dx

neg_number:
 xor ax, 0×ffff
 xor dx, 0×ffff

 ret

#; rdi -- string to write
display_dx_ax:
 push rdi
 push rsi
 push rdx
 push rcx

 xor rcx, rcx

 mov rdi, offset dx_ax_str
```

```
 push rax
 mov  cx, 4
display_dx_ax_proceed_dx_space:
 push cx
 mov  cx, 4
display_dx_ax_proceed_dx:
 rcl  dx, 1
 jc   proceed_dx_one

 mov  ax, '0'
 jmp  proceed_dx_next

proceed_dx_one:
 mov  ax, '1'

proceed_dx_next:
 stosb
 loop display_dx_ax_proceed_dx

 pop  cx
 mov  ax, ' '
 stosb

 loop display_dx_ax_proceed_dx_space
 pop  rax

 mov  dx, ax
 push rax
 mov  cx, 4
display_dx_ax_proceed_ax_space:
 push cx
 mov  cx, 4
display_dx_ax_proceed_ax:
 rcl  dx, 1
 jc   proceed_ax_one

 mov  ax, '0'
 jmp  proceed_ax_next

proceed_ax_one:
 mov  ax, '1'

proceed_ax_next:
 stosb
 loop display_dx_ax_proceed_ax

 pop  cx
 mov  ax, ' '
 stosb
```

```
 loop display_dx_ax_proceed_ax_space

 mov  rax, 1
 mov  rdi, 1
 mov  rsi, offset dx_ax_str
 mov  rdx, dx_ax_str_len
 syscall

 mov  rax, 1
 mov  rdi, 1
 mov  rsi, offset newline_str
 mov  rdx, 1
 syscall

 pop  rax

 pop  rcx
 pop  rdx
 pop  rsi
 pop  rdi
 ret

.global  _start
_start:
#; Insert number
 mov  edx, input_number

 mov  ax, dx
 shr  edx, 16

#; Display dx:ax registers
 call display_dx_ax

 rcl  dx, 1
 jnc  positive
 rcr  dx, 1
 call invert_number
 jmp  make_number_str

positive:
 shr  dx, 1

make_number_str:
 call make_str

#; Print oct number
 mov  rax, 1
 mov  rdi, 1
 mov  rsi, offset number_str
 mov  rdx, number_str_len
 syscall
```

```
#; Print new line
 mov  rax, 1
 mov  rdi, 1
 mov  rsi, offset newline_str
 mov  rdx, 1
 syscall

#; Get value from str to dx:ax
 mov  rsi, offset number_str
 call get_number

#; Display dx:ax registers
 call display_dx_ax

#; Exit
 mov  rax, 60
 mov  rdi, 0
 syscall
```

Название файла: Makefile

```
all: main

main: main.o
	ld main.o -o main

main.o: main.s
	as main.s -msyntax=intel -mmnemonic=intel -mnaked-reg -o main.o

clean:
	rm -f *.o main
```