МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Организация ЭВМ и систем»

Тема: Использование арифметических операций над целыми числами и процедур в Ассемблере.

Студент гр. 9382 _____ Сорокумов С.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить арифметические операции над целыми числами и процедуры в Ассемблере.

Теоретические сведения:

Разработать на языке Ассемблер процессора IntelX86 две процедуры:

• одна – выполняет прямое преобразование целого числа, заданного в регистре AX (или в паре регистров DX:AX) в строку, представляющую его символьное изображение в заданной системе счисления (с учетом или без учета знака в зависимости от варианта задания);

• другая - обратное преобразование строки, представляющей символьное изображение числа в заданной системе счисления в целое число, помещаемое в регистр AX (или в пару регистров DX:AX)

Строка должна храниться в памяти, а также выводиться на экран для индикации.

Отрицательные числа при представлении с учетом знака должны в памяти храниться в дополнительном коде, а на экране изображаться в прямом коде с явным указанием знака или в символьном виде со знаком.

Задание:

Вариант 1.2.4 – 16 бит, без учета знака, 16-ичная СС

Ход работы:

В регистр записывается число, которое необходимо перевести в строку. В программе реализовано две процедуры DEC_TO_HEX и HEX_TO_DEC.

Первая процедура DEC_TO_HEX начинает с проверки знака числа. Если число отрицательное, то оно инвертируется и инкрементируется для

того, чтобы корректно был произведен перевод в строку. В начале строки записывается знак. Если число равно нулю, то сразу записывается нуль, как ответ, так как в последующем коде такой случай был бы исключительным. Ранее была объявлена переменная, нужная для того, чтобы проследить, нужно ли в строку записывать спереди идущие нули. Программа из числа берет цифру и записывает в символьном виде в строку, проверяя переменную, которая была упомянута в прошлом предложении. В конец строки добавляется символ конца строки, и строка выводится.

Вторая процедура HEX_TO_DEC происходит считывание количества цифр и проход по строке. Расстояние в таблице ASCII между цифрами и буквами, использующимися в 16-ирчной СС равно 7, поэтому случаи с буквами нужно рассматривать немного по-другому. Результат записывается в другой регист для удобной работы, затем возвращается в AX.

В основной процедуре сначала вызывается DEC_TO_HEX и выводится строка, которая является числом в 16-ичной СС. Затем вызывается HEX_TO_DEC, для проверки корректности вызывается заново DEC_TO_HEX, если строки совпадают, то их корректность очевидна.

Тестирование.

№	Входные данные	Результат
1.	AX = 0h	Transformation to string: +0 Transformation from string to digit and back: +0
2.	AX = 1h	Transformation to string: 1 Transformation from string to digit and back: 1
3.	AX = 801h	Transformation to string: 801 Transformation from string to digit and back: 801
4.	AX = 9999h	Transformation to string: 9999 Transformation from string to digit and back: 9999

Выводы.

В результате выполнения лабораторной работы был разработан код, использующий арифметические операции над целыми числами и процедуры в ассемблере, написаны две процедуры по преобразованию числа в строку и обратно.

Приложение.

Текст файла 7lab.asm

```
STACKSG SEGMENT  PARA STACK 'Stack'
        DW       512 DUP(?)
STACKSGENDS

DATASG  SEGMENT  PARA 'Data'; SEG DATA
KEEP_CS DW 0 ;
        MESSAGE1 DB 'Transformation to string: $'
        MESSAGE2 DB 'Transformation from string to digit and back: $'
STRING DB 35 DUP('0')
DATASG ENDS; ENDS DATA


CODE SEGMENT; SEG CODE
ASSUME  DS:DataSG, CS:Code, SS:STACKSG
;0…+65 535

DEC_TO_HEX PROC NEAR
jmp start
delete_nul DW 0; нужна для того, чтобы не писать впереди стоящие нули
start:
mov delete_nul, 0
mov DI, 0h; DI - индекс текущего символа строки
cmp AX, 0; если число равно нулю, то сразу пишем нуль
je case_nul
jmp scan_ax;обрабатываем число

check_nul:  ;служит для определения необходимости записи нуля
cmp delete_nul, 0 ;если нуль значащий, то записиваем
je skip_char
jne no_skip_char

scan_ax:
mov SI,AX ; записываем в si, ax
mov   cx, 4        ; в слове 4 ниббла (полубайта)

next_char:
rol   ax, 1        ; выдвигаем младшие 4 бита
rol   ax, 1
rol   ax, 1
rol   ax, 1
push  ax           ; сохраним AX
and   al, 0Fh            ; оставляем 4 младших бита AL
cmp   al, 0Ah            ; сравниваем AL со значением 10
sbb   al, 69h            ;   целочисленное   вычитание   с заёмом
das                ; BCD-коррекция после вычитания
cmp al, '0' ;если нуль
je check_nul
```

```
      mov delete_nul, 1; если попалась цифра, отличная от нуля,
то остальные нули будут значащими

      no_skip_char:
      mov STRING[DI], al ;записываем число в строку
      add DI, 1; инкрементируем счетчик

      skip_char:
      pop    ax           ; восстановим AX для следующих цифр
      loop next_char
      jmp end_1

      case_nul:
      mov STRING[DI], '0'
      add DI, 1

      end_1: ; когда прошли все регистры
      mov  STRING[DI],'$'  ;  добавляем  в  конец  строки  символ
конца строки
      mov DX,offset STRING ; записываем в dx сдвиг строки
      ret
      DEC_TO_HEX ENDP

      HEX_TO_DEC PROC FAR

      mov AX,0; обнуляем ax и cx
      mov CX, 0
      mov SI,0; за индекс строки будет отвечать si

      len_loop: ; считаем длину строки
      add SI, 1
      cmp STRING[SI],'$' ; сравниваем элемент строки с $
      jne len_loop ; если не равен $ то возвращаемся в цикл

      mov  DI,  SI;  в  di  будет  храниться  количество  цифр  в
числе
      lea SI, STRING; будем работать со строкой
      xor cx, cx
      add DI, 1
      cld

      number_construct:
      xor AX, AX
      dec DI ; декремингим DI
      cmp DI,0 ; сравниваем DI с 0
      jle done ; DI <= 0 заканчиваем обработку строки
      lodsb; в al кладется очередной символ
      cmp al, 'A'
      jge bukva ; если больше или ровно

      continue:
      sub al, '0' ;работаем с цифрой вместо кода цифры
      xchg  ax,  cx;  меняем  значения,  так  как  в  cx  лежит
результат
      mov dx, 10h
          mul dx; ax * 10
```

```
	add cx, ax; прибавляем в прошлому результату следующую цифру
	jmp number_construct

	done:
	mov ax, cx; в ax кладем результат
	jmp end_2

	bukva:
	sub al, 7; убираем разрыв между цифрами и буквами
	jmp continue

	end_2:
	ret
	HEX_TO_DEC ENDP


	Main PROC FAR
		mov  ax, DATASG
		mov  ds, ax

		 mov DX, offset MESSAGE1 ;вывод первого сообщения
		 mov ah,09h;
	int 21h;
	mov AX, 0801h ;наше число
	call DEC_TO_HEX
	mov ah,09h ;вывод строки
	int 21h;

	mov dl, 10;возврат каретки
	mov ah, 02h
	int 21h
	mov dl, 13; новая строка
	mov ah, 02h
	int 21h

		 mov DX, offset MESSAGE2 ;вывод второго сообщения
		 mov ah,09h;
	int 21h;
	mov ax, 0
	call HEX_TO_DEC
	call DEC_TO_HEX

	mov ah,09h ;вывод строки
	int 21h

	mov ah,4Ch;завершение
	int 21h;

	Main	ENDP
	CODE	ENDS
	END Main
```