4. ЛАБОРАТОРНАЯ РАБОТА № 4 «МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ»

Цель работы: знакомство с технологией применения языка ассемблера при разработке программного обеспечения на языках высокого уровня.

1. Многомодульный проект

Для смешанного программирования на языке высокого уровня и ассемблере прекрасно подходит Си. Для объединения кода ассемблера и Си в нем предусмотрены:

- встроенный ассемблер;
- многомодульный проект.

Создание проекта с использованием модулей C++ и ассемблера имеет ряд особенностей, связанных с выбором имен переменных и функций.

Так как транслятор языка C++ имеет встроенный режим генерации имен (из-за поддержки перегрузки функций), связь модулей C++ и ассемблера выполняется обычно через функции Си.

Чтобы скомпоновать вместе модули Си и ассемблера, должны быть соблюдены следующие условия:

- в модулях ассемблера должна использоваться схема наименования сегментов и модель памяти, совместимая с Си;
- Си и ассемблер должны совместно использовать соответствующие функции и имена переменных в форме, необходимой для Си;
- глобальные переменные, объявленные в модулях языка Си, должны иметь описание в ассемблерном модуле в сегменте данных (.DATA) с помощью директивы EXTERN;
- функции языка Си, вызов которых осуществляется из модулей ассемблера, должны иметь прототип или описание с помощью директивы EXTERN в сегменте кода (.CODE).

1.1 Модели памяти и сегменты

Чтобы функция ассемблера могла вызываться из Си, она должна использовать ту же модель памяти, что и программа на языке Си, а также совместимый с Си сегмент кода. В настоящее время 32-разрядные приложения используют модель FLAT – плоская модель. Для этой модели характерны 32-разрядные указатели для переменных и 32-разрядные адреса для функций (дальний вызов). В директиве модели памяти для совместимости с языками высокого уровня можно использовать выбор языка – C, Pascal, Fortran, Prolog и др. Выбор языка устанавливает дисциплину объявления имен переменных и функций, а также определяет правила вызова и возврата из подпрограмм. Например, директива

model flat , C

устанавливает модель памяти FLAT, а также соглашения по именованию переменных и функций и правила передачи параметров подпрограммам и возврату результата из подпрограмм согласно правилам языка Си.

1.2 Общедоступные и внешние переменные

Следует иметь в виду, что транслятор языка Си добавляет к именам переменных и функций дополнительный символ «_». При использовании встроенного ассемблера можно обращаться к переменным и функциям без указания этого дополнительного символа.

Программы ассемблера могут вызывать функции Си и ссылаться на внешние переменные Си. Программы Си аналогичным образом могут вызывать общедоступные (PUBLIC) функции ассемблера и обращаться к переменным ассемблера.

В табл. 1 представлено соответствие между типами данных Си и ассемблера. Для доступа в модулях на языке ассемблера к переменным, объявленным в модулях на языке Си, используется директива EXTERN. Директива размещается в области данных (.DATA).

Табл. 1. Соответствие типов данных

Тип данных Си	Тип ассемблера	Примечание
unsigned char	BYTE	Символ
char	BYTE	-//-
enum	WORD	Перечислитель
unsigned short	WORD	Короткое целое без знака
short	WORD	Короткое целое со знаком
unsigned int	DWORD	Целое без знака
int	DWORD	Целое со знаком
unsigned long	DWORD	Длинное целое без знака
long	DWORD	Длинное целое со знаком
float	DWORD	Вещественное
double	QWORD	Вещественное двойной точности
long double	TBYTE	Расширенной точности
far *	DWORD	Дальний указатель

Директива EXTERN имеет синтаксис

EXTERN имя : тип

Имя переменной должно начинаться с символа «_». Тип выбирается в соответствии с табл. 1. Например, если переменные X, y и zi были объявлены как short, int и int * соответственно, то для доступа к ним в модуле на языке ассемблер должно иметься описание:

```
. MODEL FLAT,C
.DATA
EXTERN _X : word
EXTERN _y : dword
EXTERN _zi : dword
```

Для доступа в модулях на языке Си к переменным, объявленным в модулях на языке ассемблера, используется директива PUBLIC. Директива размещается в области данных (.DATA).

Синтаксис директивы PUBLIC

PUBLIC имя ,

где имя – имя переменной.

Например, если переменная RAB, Ob, Sec на языке ассемблера объявлены размером DB, DW и DD соответственно, то в модуле на языке ассемблера будет иметься описание:

```
. MODEL FLAT
.DATA
PUBLIC _RAB,_ Ob, _Sec
_RAB DB ‘1’
_Ob DW 23
_Sec DD ? ,
```

а в модуле на языке C++

```
extern "C" char RAB;
extern "C" short Ob;
extern "C" int Sec;
```

Использование директивы extern "C" требуется для отмены режима генерации имен для языка C++.

1.3 Функции и передача параметров в функцию

Так как транслятор языка C++ имеет режим генерации имен, обращение к функциям C++ зачастую невозможно (если правила генерации имен не известны). В таких случаях используют переходник (дополнительная функция), поддерживающий соглашение языка Си (рис. 1).

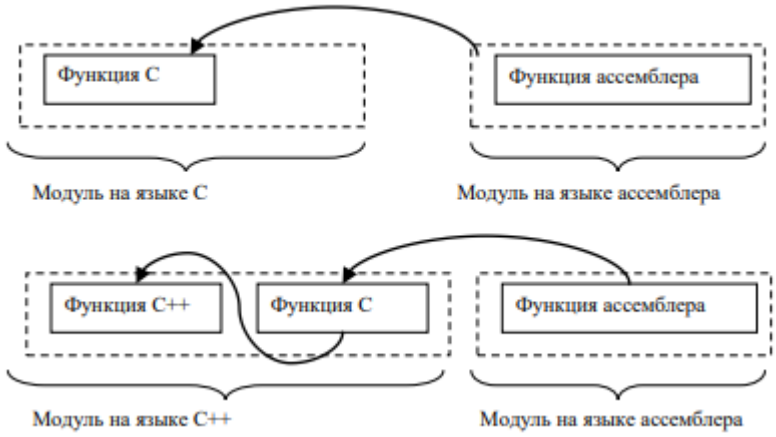

Рис.1. Организация вызова функций с использованием языка Си/C++ и ассемблера

Для доступа в модулях на языке ассемблера к функциям, объявленным в модулях на языке Си, используется директива EXTERN. Директива размещается в области кода (.CODE).

Директива EXTERN имеет синтаксис

EXTERN имя : тип

Тип выбирается в соответствии с табл. 1.1 для указателей (near, far). Следует иметь в виду, что по умолчанию все функции языка С имеют тип FAR для 32-х разрядных и NEAR – для 16-ти разрядных предложений.

Для доступа в модулях на языке Си к функциям, объявленным в модулях на языке ассемблера, используется директива PUBLIC. Директива размещается в области кода (.CODE). Директива PUBLIC имеет синтаксис

PUBLIC имя

В модуле на языке Си должен быть прототип функции, описывающий функцию как внешнюю.

```
extern "C" int fib(int a);
```

Тип возврата, а также количество и типы аргументов не проверяются. Вся ответственность за согласование параметров функции и возвращаемого значения возлагается на разработчика.

Аналогично осуществляется взаимодействие при вызове подпрограмм ассемблера со стороны модулей языка Си (C++).

1.4. Передача параметров

Си передает функциям параметры через стек. Перед вызовом функции Си сначала заносят в стек передаваемые этой функции параметры, начиная с самого правого параметра и заканчивая левым. В Си вызов функции:

```
Test(i, j, 1);
```

компилируется в инструкции:

```
mov eax,1
push eax
push _j
push _i
```

```
call _Test
add esp,12
```

где видно, что правый параметр (константа 1) заносится в стек первым, затем туда заносится параметр j и, наконец, i.

При возврате из функции параметры, занесенные в стек, все еще находятся там, но они больше не используются. Поэтому непосредственно после каждого вызова функции требуется установить в указателе стека значение, которое он имел перед занесением в стек параметров.

В предыдущем примере три параметра (по четыре байта каждый) занимают в стеке вместе 12 байт, поэтому команда add esp,12 добавляет значение 12 к указателю стека, чтобы отбросить параметры после обращения к функции Test. По соглашениям Си за удаление параметров из стека отвечает вызывающая программа.

Функции ассемблера могут обращаться к параметрам, передаваемым в стеке, относительно регистра BP (EBP). Например, предположим, что функция Test в предыдущем примере представляет собой следующую функцию на Ассемблере:

```
DOSSEG
.MODEL FLAT
.CODE
PUBLIC _Test
_Test PROC
    push ebp
    mov ebp,esp
    mov eax,[bp+4] ; получить параметр 1
    add eax,[bp+8] ; прибавить параметр 2
    sub eax,[bp+12] ; вычесть из суммы 3
    pop ebp
    ret
_Test ENDP
```

Как можно видеть, функция Test получает передаваемые из программы на языке Си параметры через стек относительно EBP. На рис.2 показано, как выглядит стек перед выполнением первой инструкции в функции Test:

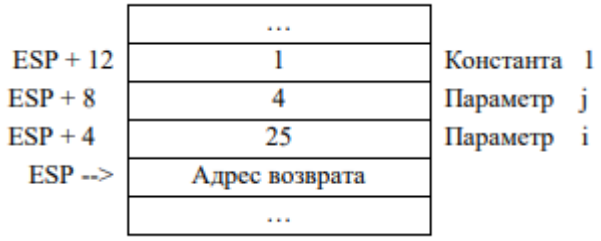

Рис.2. Состояние стека перед выполнением первой команды функции Test

Параметры функции Test представляют собой фиксированные адреса относительно ESP. Обычно для работы с параметрами используется регистр EBP. Две команды функции осуществляют сохранение предыдущего значения регистра EBP и задание нового значения:

```
push ebp
mov ebp,esp
```

Значение регистра EBP при выполнении команд функции не изменяется. Смещения до области хранения значений параметров функции представлены на рис.3.

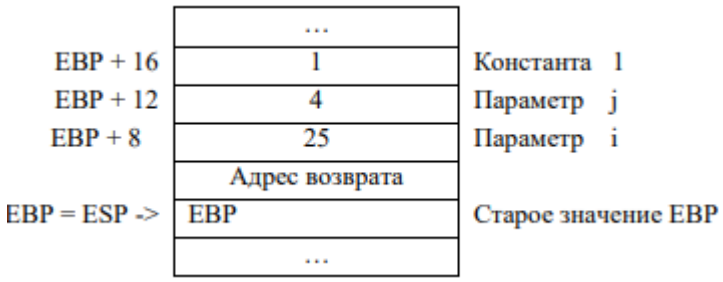

Рис. 3. Состояние стека после инструкций PUSH и MOVE

1.5. Возврат значений

Вызываемые из программы на языке Си функции на ассемблере, так же, как и функции Си, могут возвращать значения. Значения функций возвращаются в ресурсах процессора (представлено в табл. 2).

Табл. 2. Возвращение значений из функций

Тип возвращаемого значения	Регистр
unsigned char, char	AL(AX)
unsigned short, short, enum	AX
unsigned int, int	EAX
unsigned long long	EAX
Float	регистр сопроцессора 80x87 ST(0)
Double	регистр сопроцессора 80x87 ST(0)
long double	регистр сопроцессора 80x87 ST(0)
near*	AX
far*	EAX

1.6 Сохранение регистров

При взаимодействии ассемблера и Си функции ассемблера, вызываемые из программы на языке Си, могут использовать любые регистры, но при этом они должны сохранять регистры EBP, ESP, CS, DS и SS. Содержимое этих регистров можно изменять и/или использовать для промежуточных расчетов. Перед возвратом из вызываемой подпрограммы регистры EBP, ESP, CS, DS и SS должны иметь в точности такие значения, какие они имели при ее вызове.

Регистры EAX, EBX, ECX, EDX, а также флаги могут произвольно изменяться. Содержимое этих регистров можно не сохранять.

Регистры EDI и ESI представляют собой особый случай, так как в Си они могут использоваться для регистровых переменных. Если в модуле Си, из которого вызывается функция на ассемблере, использование регистровых переменных разрешено, то содержимое регистров нужно сохранить.

2. Пример разработки приложения

Пусть необходимо разработать приложения для подсчета суммы

$$S = \sum_{k=0}^{n} \frac{\sqrt[3]{\ln(x_i)}}{i}.$$

Реализуем подсчёт суммы с использованием подпрограммы на языке ассемблера, а расчет элемента суммы – с помощью функции на языке С.

Головной модуль будет написан с использованием языка C++. Схема вызовов подпрограммы ассемблера и функции C представлена на рис.4.

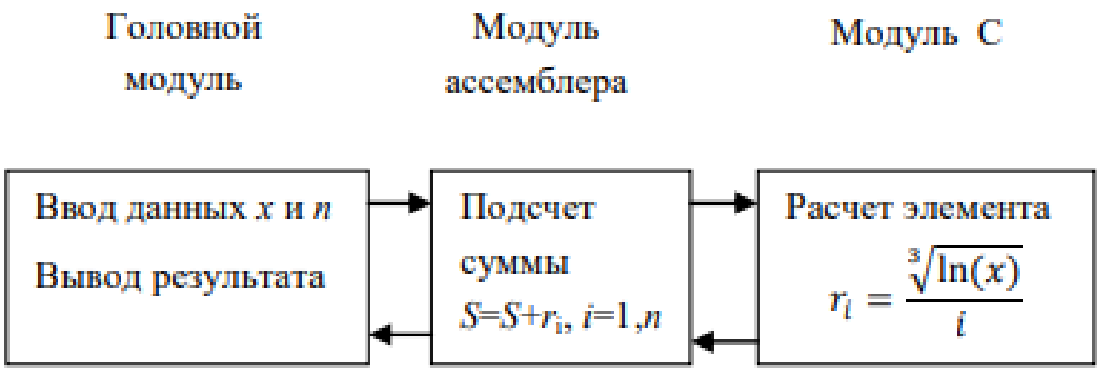

Рис. 4. Схема вызовов модулей приложения

Головной модуль

```
#include <iostream>
using namespace std;
extern "C" float SumR(int k, float x);
int main (int argc,char ** argv)
{
    int n;
    float x;
    cout << "Input x ,n" << endl;
    cin >> x >> n;
    double R = 0.0;
    R = SumR(n, x);
    cout << "Result" << R << endl;
    return 0;
}
```

Модуль ассемблера содержит функцию SumR. Функция имеет два параметра – число элементов ряда и значение переменной x. Функция использует две локальные статические переменные: SUM () и i_local (номер элемента ряда). Регистр ECX используется для организации цикла. Перед вызовом функции fun_el содержимое регистра ECX необходимо сохранить и восстановить после возврата.

```
.586 .MODEL flat,C
.DATA
SUM DD 0.0
i_local DD 0
.CODE
extern fun_el:near ; объявление внешней функции fun_el
public SumR
SumR proc C
	push ebp
	mov ebp,esp
	mov i_local,1
	mov ecx, dword ptr [ebp+8]
	@@for_i: ; начало цикла
		push ecx
		push dword ptr [ebp + 12]
		push i_local
		call fun_el
		fld SUM
		add esp,8
		fadd
		pop ecx
		inc i_local
		fstp SUM
	loop @@for_i ; конец цикла
	fld SUM
	mov esp,ebp
	pop ebp
	ret
	SumR endp
End
```

Модуль на языке С содержит функцию для расчёта элемента ряда.

```
Extern "C" float fun_el (int i, float z)
{ float f;
 f= powf ( log10f(z)/log10f(2.71828), 1.0f/3.0f ) / (float) i;
return f; }
```

2. Индивидуальное задание

1. Сформировать проект консольного приложения с использованием модуля на основе языка ассемблера.

2. Разработать основной модуль приложения. Основной модуль обеспечивает ввод данных, вызов подпрограммы ассемблера и вывод результата.

3. Добавить в проект модуль на языке ассемблера. Разработать подпрограмму на языке ассемблера в соответствии с вариантом (табл. 3).

4. Добавить в проект модуль на языке Си. Разработать функцию на языке высокого уровня в соответствии с вариантом.

5. Обеспечить вызов подпрограммы ассемблера из основного модуля на языке С++. Основной модуль обеспечивает ввод данных и вывод результата.

6. Осуществить вызов функции языка Си из модуля ассемблера.

7. Проверить работу приложения в режиме отладки. Записать содержимое стека перед вызовом подпрограммы ассемблера и функции Си.

Табл. 3. Варианты заданий

Вариант	Функция языка Си	Подпрограмма ассемблера
1	Вычисление значения элемента ряда $a_k = \frac{x^{-k}}{\sin(kx)}$	Вычисление суммы ряда $z = \sum_{k=0}^{n} a_k$
2	Вычисление значения функции $f(x) = \frac{\cos(x) + \sin(x)}{e^x}$	Вычисление $\max(f(x))$ x= –1 ÷ +2

3	Вычисление значения функции $f(x) = \sqrt[3]{\text{tg}(x)}$	Вычисление определенного интеграла $y = \int_0^1 f(x)$
4	Вычисление значения функции $f(x) = \frac{\text{tg}(x) + \sin(x)}{e^x}$	Вычисление значений функции $y_i = f(x_i)$ на интервале 0 ÷ 1,шагом 0,1
5	Вычисление значения функции $f(x) = \lg(\lvert cox(x) - \sin(x)\rvert)$	Вычисление значений функции $y_i = f(x_i) \mid i = 1, n$
6	Вычисление значения элемента ряда $a_k = \frac{x^{-k}}{\text{k!}\cos(x)}$	Вычисление суммы ряда $z = \sum_{k=0}^{n} a_k$

7	Вычисление значения функции $f(x) = \frac{\text{tg}(x) + \sin(x)}{\cos(x)}$	Вычисление $\min(f(x))$ x= –1 ÷ +1
8	Вычисление значения функции $f(x) = \sqrt[3]{\ln(x)}$	Вычисление определенного интеграла $y = \int_0^1 f(x)$
9	Вычисление значения функции $f(x) = \frac{\text{tg}(x) + \sin(x)}{e^x}$	Вычисление значений функции $y_i = f(x_i)$ на интервале 0 ÷ 1 с шагом 0,1
10	Вычисление значения функции $f(x) = (\lvert ctg(x) - \sin(x)\rvert)$	Вычисление значений функции $y_i = f(x_i) \mid i = 1, n$
11	Вычисление значения элемента ряда $a_k = \frac{x^{-1/k}}{\sin(x)}$	Вычисление суммы ряда $z = \sum_{k=0}^{n} \lvert a_k \rvert$

12	Вычисление значения функции $f(x) = \frac{\cos(x) - 2 \times \sin(x)}{e^x}$	Вычисление $\min(f(x))$ x= –1 ÷ +1, шаг 0,01
13	Вычисление значения функции $f(x) = \sqrt[3]{\text{ctg}(x)}$	Вычисление определенного интеграла $y = \int_0^1 \lvert f(x) \rvert$
14	Вычисление значения функции $f(x) = \frac{\text{ctg}(x) + cos(x)}{e^x}$	Вычисление значений функции $y_i = f(x_i)$ на интервале 0 ÷ 1,шаг 0,1
15	Вычисление значения функции $f(x) = \ln(\lvert cox(x) + \sin(x) \rvert)$	Вычисление значений функции $y_i = f(x_i) \mid i = 1,n$
16	Вычисление значения элемента ряда $a_k = \frac{x^{-k}}{\text{k!}\cos(x)}$	Вычисление суммы ряда $z = \sum_{k=0}^{n} \lvert a_k \rvert$
17	Вычисление значения функции $f(x) = \frac{\text{ctg}(x) + \sin(x)}{\sin(x)}$	Вычисление $\min(f(x))$ x= –1 ÷ +1
18	Вычисление значения функции $f(x) = \lvert \sqrt[3]{\ln(x)} \rvert$	Вычисление определенного интеграла $y_i = f(x_i) \mid i = 1,n$
19	Вычисление значения функции $f(x) = \frac{\text{tg}(x) + \sin(x)}{e^x}$	Вычисление значений функции $y_i = f(x_i)$ на интервале 0 ÷ 1, шаг 0,1
20	Вычисление значения функции $f(x) = (\lvert ctg(x) + \sin(x) \rvert)$	Вычисление значений функции $y = \int_0^1 f(x)$

3. Содержание отчета:

1. Текст программы с комментариями

2. Верификация программы: описание и пример решения задачи, скриншоты, показывающие содержимое регистров и значения переменных после каждого

действия программы, входные данные и скриншоты регистров и переменных в ключевых точках программы для проверки работы программы при разных входных данных.