3 семестр. [ЭУМК] Архитектура компьютеров

Настройки

Управление курсом

Настройки моего профиля

Навигация

В начало

- Моя домашняя страница

Страницы сайта

Мой профиль

Текущий курс

ПИ 3. АК

Участники

ЭУМК «Архитектура компьютеров»

Модуль 1. «Базовые принципы архитектуры»

Модуль 2. «Основные архитектурные решения»

6. Арифметический сопроцессор

7. Архитектуры многоядерных процессоров

Лабораторная работа №8. Косвенный вызов подпрограмм в языке Ассемблера

Лабораторная работа №8

Косвенный вызов подпрограмм в языке Ассемблера

Задание 1. Изучите примеры описания и использования структур, а также примеры косвенного вызова подпрограмм. Напишите программу, демонстрирующую эти примеры. Описание работы со структурами в языке Ассемблера смотри ниже.

Пример описания структуры и объявления массива:

```
; описание структуры для хранения комплексного числа
complex struct
        re dq ?
        im dq ?
complex ends

.data
        ; объявление массива из 5-ти комплексных чисел
        array complex <1.0, 2.0>, <-4.5, 1.25>, <0.0, -3.1>, <3.5, -1.5>, <
```

Пример подпрограммы, сравнивающих два комплексных числа, переданных в подпрограмму по адресу соответствующих ячеек памяти. Подпрограмма сравнивает только вещественные части двух комплексных чисел и возвращает в регистре EAX число: -1, если первый аргумент меньше второго; +1, если первый аргумент больше второго; 0, если аргументы равны:

```
complexCompareByRealPart:
        ; получаем адрес первого аргумента
        mov EAX, [ ESP + 4 ]
        ; заносим в вершину стека сопроцессора
        ; действительную часть первого аргумента
        fld qword ptr [ EAX ]
        ; получаем адрес второго аргумента
        mov EAX, [ ESP + 8 ]
        ; сравниваем действительную часть первого
        ; аргумента из вершины стека сопроцессора
        ; с действительной частью второго аргумента,
        ; находящегося в памяти; при этом число из
        ; вершины стека сопроцессора удаляется
        fcomp qword ptr [ EAX ]
        ; копируем флаги сопроцессора в регистр
        ; флагов основного процессора
        fstsw AX
        sahf
        ; возвращаем необходимый целочисленный результат
        ja great1
        jb less1
                mov EAX, 0
                jmp return1
        great1:
                mov EAX, 1
                jmp return1
        less1:
                mov EAX, -1
        return1:
ret 8
```

Пример подпрограммы, сравнивающих два комплексных числа, переданных в подпрограмму по адресу соответствующих ячеек памяти. Подпрограмма сравнивает модули двух комплексных чисел и возвращает в регистре EAX число: -1, если первый аргумент меньше второго; +1, если первый аргумент больше второго; 0, если аргументы равны:

```
complexCompareByModulus:
        ; получаем адрес первого аргумента
        mov EAX, [ ESP + 4 ]
        ; заносим в вершину стека сопроцессора
        ; действительную часть первого аргумента
        fld qword ptr [ EAX ]
        ; возводим действительную часть первого
        ; аргумента в квадрат
        fmul ST (0), ST (0)
        ; заномив в вершину стека сопроцессора
        ; мнимую часть первого аргумента
        fld qword ptr [ EAX + 8 ]
        ; возводим мнимую часть первого аргумента
        ; в квадрат
        fmul ST (0), ST (0)
        ; складываем в регистре ST (1) квадрат
        ; действительной и мнимой части, после
        ; второй операнд ST (0) удаляется из стека,
        ; а результат подымается в вершину стека
        faddp ST (1), ST (0)
        ; производим вычисление модуля второго
        ; комплексного числа аналогично первому
        mov EAX, [ ESP + 8 ]
        fld qword ptr [ EAX ]
        fmul ST (0), ST (0)
        fld qword ptr [ EAX + 8 ]
        fmul ST (0), ST (0)
        faddp ST (1), ST (0)
        ; находим разность квадратов модулей
        ; комплексных чисел
        fsubp ST (1), ST (0)
        ; сравнение разности модулей с нулём
        ficomp zero
        fstsw AX
        sahf
        ; возвращаем целочисленный результат
        ja great2
        jb less2
                mov EAX, 0
                jmp return2
        great2:
```

```
                mov EAX, 1
                jmp return2
        less2:
                mov EAX, -1
        return2:
ret 8
```

Пример поиска адреса максимального элемента массива, при этом сравнение элементов производится с помощью некоторой функции, адрес которой передаётся в данную подпрограмму:

```
max:
        ; первый параметр - адрес первого
        ; элемента массива
        mov EBX, [ ESP + 4 ]
        ; количество элементов массива
        mov ECX, [ ESP + 8 ]
        ; адрес функции, которая будет
        ; сравнивать два элемента массива
        mov EDX, [ ESP + 12 ]

        ; заносим в регистр ESI адрес
        ; первого элемента массива, в
        ; дальнейшем здесь будем хранить
        ; адрес макстмального элемента
        mov ESI, EBX

        beginCycle:
                ; проверяем, достигнут ли
                ; конец массива
                cmp ECX, 0
                je endCycle
                ; передаём в функцию сравнения
                ; адрес текущего максимального
                ; элемента массива
                push ESI
                ; передаём в функцию сравнения
                ; адрес очередного элемента массива
                push EBX
```

```
                ; вызываем функцию сравнения,
                ; используя концепцию косвенного вызова
                call EDX
                ; сравниваем результат, возвращённый
                ; функцией сравнения, с нулём
                cmp EAX, 0
                ; если текущий максимальный элемент
                ; меньше очередного элемента массива
                jng skip
                        ; запоминаем в качестве адреса
                        ; максимального элемента массива
                        ; адрес текущего элемента массива
                        mov ESI, EBX
                skip:
                ; переходим к следующему элементу массива
                ; (используем смещение 16 байт - размер
                ; структуры complex)
                add EBX, 16
                ; уменьшаем количество элементов массива
                dec ECX
                jmp beginCycle
        endCycle:
        ; заносим в регистр EAX адрес максимально
        ; элемента массива
        mov EAX, ESI
ret 12
```

Пример вызова подпрограммы `max` для поиска максимального элемента в массиве комплексных чисел, используя сравнение действительных частей элементов:

```
push complexCompareByRealPart
push 5
push offset array
call max
```

Пример вызова подпрограммы `max` для поиска максимального элемента в массиве комплексных чисел, используя сравнение моулей элементов:

```
push complexCompareByModulus
push 5
push offset array
call max
```

Теория:

Для объявления структур в языке Ассемблера используются ключевые слова `struct` (начало объявление структуры) и `ends` (конец объявления структуры). Перед каждым из этих ключевых слов указывается имя структуры. Пример:

```
имя_структуры struct
        ; описание полей структуры
имя_структуры ends
```

При описании полей структуры используются директивы объявления переменных, такие же, что и в сегментах инициализированных и неинициализированных данных. При этом те поля структуры, которые объявлены, как инициализированные данные, имеют некоторое значение по умолчанию.

Объявлять переменную типа структуры можно как в сегменте `.data`, так и в сегменте `.data?`. В обоих случаях структуру нужно инициализировать явно, используя скобки вида <...>, например:

```
имя_переменной имя_структуры <
        значение_первого_поля структуры,
        значение_второго_поля_структуры,
        ...,
        значение_N_го_поля структуры
>
```

При этом, в случае описания неинициализированной структуры, вместо значения полей структуры используется знак `?`. При этом, если поле структуры при объявлении было проинициализировано, то при описании переменной соответствующего типа это поле примет значение по умолчанию (указанной в объявлении). В ином случае значение поля будет считаться непроинициализированным.

Для обращения к полям структуры можно использовать специальный оператор `.`, например:

```
имя_переменной.имя_поля
```

или же используя адрес начала структуры и смещение полей структуры (зная размер каждой структуры в байтах).

Задание 2. Напишите программу, выводящую список некоторых элементов, отсортированный с использованием указанного алгоритма сортировки (см. свой вариант) по двум критериям (сначала вывести список, отсортированный по одному критерию; затем вывести тот же список, отсортированный по второму критерию). Алгоритм сортировки реализовать независимо от используемого критерия сортировки, используя концепцию косвенного вызова.

Варианты:

1. Отсортировать методом выбора список результатов замеров массы тела спортсмена (день, месяц, год взвешивания, результат взвешивания - дробное число) по массе и по дате взвешивания.
2. Отсортировать методом выбора список результатов замеров температуры телы больного (часы, минуты, температура - дробное число) по значению температуры и по времени измерения.
3. Отсортировать методом выбора список правильных многоугольников (количество сторон, длина стороны - дробное число) по площади и по периметру.
4. Отсортировать методом выбора список билетов в кинотеатр (ряд, место, стоимость - дробное число) по стоимости и по ряду с местом.
5. Отсортировать методом выбора список запросов к серверу (IP-адрес в формате XXX.XXX.XXX.XXX, где XXX - числа в диапазоне от 0 до 255; частота запросов в процентах - дробное число) по частоте запросов и по IP-адресу.
6. Отсортировать методом обмена список результатов замеров массы тела спортсмена (день, месяц, год взвешивания, результат взвешивания - дробное число) по массе и по дате взвешивания.

7. Отсортировать методом обмена список результатов замеров температуры телы больного (часы, минуты, температура - дробное число) по значению температуры и по времени измерения.
8. Отсортировать методом обмена список правильных многоугольников (количество сторон, длина стороны - дробное число) по площади и по периметру.
9. Отсортировать методом обмена список билетов в кинотеатр (ряд, место, стоимость - дробное число) по стоимости и по ряду с местом.
10. Отсортировать методом обмена список запросов к серверу (IP-адрес в формате XXX.XXX.XXX.XXX, где XXX - числа в диапазоне от 0 до 255; частота запросов в процентах - дробное число) по частоте запросов и по IP-адресу.

Состояние ответа

Состояние ответа на задание	Ответ на задание должен быть представлен вне сайта
Состояние оценивания	Не оценено