

Лабораторная работа asm4 (2 часа)

Конструирование программного обеспечения

Введение в Ассемблер

Цель работы: подготовка к построению плана памяти для размещения данных с мета-информацией.

Построение развернутого плана памяти данных.

Память данных – это ячейки памяти. Ячейки памяти хранят тип переменной или литерала и непосредственно сами данные:

тип	данные
тип	данные
тип	данные

План памяти с мета-данными – это ячейки памяти, которые хранят дополнительную информацию о данных (тип переменной) и непосредственно сами данные.

Введем обозначения для типов данных:

0x01 – целочисленный тип

0x02 – строковый тип

Пример для целочисленных литералов. Область памяти с данными, содержащая в первом байте признак типа литерала, затем непосредственное значение литерала, указанного типа:

0x01	0x00000000
0x01	0x00000016

память для целочисленных данных
(пустая ячейка)

память для целочисленных данных
(ячейка с литералом)

Пример для строковых данных:

0x02	0x00	0x00 0x00 0x00 ...
------	------	--------------------

ячейки памяти для пустой строки
(тип, длина, ячейка со значением)

0x02	0x03	0x61 0x62 0x63 0x00 ...
------	------	-------------------------

ячейки памяти для строковых данных
(тип, длина, ячейки со строковым литералом указанной длины)

Сериализация памяти данных:

0x01	0x00000000	0x01	0x00000016	0x02	0x00	0x02	0x03	0x61 0x62 0x63
------	------------	------	------------	------	------	------	------	----------------

Сериализация данных – это представление структуры данных в виде последовательности битов и запись этой последовательности в файл в следующем виде:

<тип1>[<длина1>]<значение1><тип2>[<длина2>]<значение2>><тип3>
[<длина3>]<значение3>...

Десериализация (обратный процесс) – это восстановление начального состояния структуры данных из битовой последовательности.

Задание.

1. **Задача:** сериализовать байты данных так, чтобы по получении можно было выполнить десериализацию в аналогичные сущности на стороне получателя.
2. Создайте решение на языке C++. Первый проект решения реализует:
 - а. определяет данные фундаментальных типов в соответствии с вариантом;
 - б. строит план памяти данных с мета-информацией;
 - с. сериализует его в файл;
3. Добавить в решение второй проект на C++, который выполняет десериализацию данных из файла:
 - а. читает информацию из файла;
 - б. выполняет десериализацию данных;
 - с. генерирует исходный код на языке ассемблера – процедуру с определением данных (использовать директивы определения данных);
 - д. записывает исходный код в файл с расширением .asm.

4. Добавить в решение проект на языке ассемблера на основе сгенерированного ассемблерного кода.
 - g. выполнить приложение в режиме отладки;
 - h. проверить десериализованные данные на соответствие типов, длин и значений исходным данным первого проекта.
5. Исследуйте выполнение программы с помощью окон отладчика: регистров, памяти, окна контрольные значения, дизассемблированного кода.
6. Названия проектов должны отражать суть задачи. Проверить работоспособность приложения на нескольких значениях для каждого заданного в вашем варианте типа.

Варианты заданий:

<i>Вариант</i>	<i>Типы данных</i>
1, 9	Переменная: текстовая строка длиной 127 байтов; целочисленный литерал (4 байта)
2, 10	Переменная: bool; целочисленный литерал (2 байта)
3, 11	Переменная: int; беззнаковый целочисленный литерал (4 байта)
4, 12	Переменная: long; беззнаковый целочисленный литерал (1 байт)
5, 13	Переменная: short; беззнаковый целочисленный литерал (2 байта)
6, 14	Переменная: массив типа char, максимальная длина 10 байтов; литерал типа bool
7, 15	Переменная: char; символьный литерал
8, 16	Переменная: wchar_t; целочисленный литерал (4 байта)

Дополнительно.

Сериализовать объект типа структура.

Сериализовать массив целочисленных целых чисел.