Языки ассемблера

Лекция 1. Введение в курс

- Программы пишутся для исполнения их процессором
- Исходная программа - это описание алгоритма действий и используемых данных с помощью символических языков.
- Чем «выше» (абстрактнее) язык, тем больше в нем символических конструкций, упрощающих реализацию алгоритмов и отладку программы.
- Для исполнения процессором исходная программа должна быть преобразована с символического языка (удобного программисту) на «язык» процессора: в коды команд и коды данных. То есть, получить «исполняемую» программу

Исполняемая программа (исполняемый код)

Это последовательность кодов для процессора. Состоит из:

- кодов команд процессора («машинные коды»), реализующих алгоритм
- кодов данных (если есть в программе)

- На внешнем носителе исполняемый код размещается в файлах с расширением .exe , .com или dll.
- Исполняемый код доступен процессору для исполнения только, если он находится в системной памяти (ОЗУ, ПЗУ).
- Загрузку исполняемой программы из файла внешнего носителя в оперативную память выполняет операционная система (ОС).

«Низкоуровневые» символические языки

- «Низкий» языковый уровень - это реализация алгоритмов непосредственно командами процессора
- Символический язык для записи исходного текста программы на уровне команд процессора называют языком ассемблера (языком транслятора).
- Ассемблер (транслятор) – служебная программа для преобразования символической записи команд и данных в машинные коды для процессора.

Пример работы транслятора

Символическая запись команд в исходном тексте:	Машинный код (в hex) после трансляции:
add al, bl	8E D8
mov ah, 4ch	B4 4C
nop	90

Отличие символических языков высокого уровня и низкоуровневых (языки ассемблеров)

ЯВУ	Языки ассемблеров
Это универсальные языки: не зависят от конкретной аппаратной платформы (архитектуры процессора)	«Машинно-зависимые» языки: используется система команд конкретного процессора (семейства процессоров)
Это алгоритмические языки: имеют разнообразные языковые конструкции, библиотеки для упрощения реализации алгоритмов. Но, на объем и эффективность будущего кода автор повлиять не может	Не алгоритмические языки Реализуя алгоритм непосредственно командами процессора, можно влиять на объем кода и его быстродействие

Отличие компиляторов с ЯВУ от трансляторов с языков ассемблера

Цель: получить из исходного текста исполняемый код для процессора

- Компилятор - преобразует один оператор ЯВУ в последовательность из нескольких машинных команд:

  1 оператор → ?? машинных команд

- Транслятор преобразует одну символически записанную команду процессора в одну машинную команду:

  1 символическая команда → 1 машинная команда

Сравним объем полученного исполняемого кода компилятором и транслятором на простом примере:

Пример: Увеличить на 2 каждый элемент массива из 5-ти однобайтных чисел

Исходный текст на C++ (primer.cpp)

```
void main ()  ;
{
short int x[5] = {1,5, -3, 23, 12} ;
for (int k =0; k<5; ++k)  x[k]=
      x[k]+2;
}
```

==========

Исходный текст - 98 байт.

Исполняемый код Primer.exe, полученный компилятором Borland C++ v.3, составляет 6356 байт

Исходный текст на ассемблере (primer.asm)

```
code  segment
assume cs:code
a1:  mov bx, ds            ; команды
     mov ds, bx
     xor si,si
     mov cx,5
a2:  add cs:x[si],2
     inc si
     loop a2
     mov ah,4ch
     int 21h
x    db  1,5,-3,23,12      ; данные
code  ends
     end a1
```

==========

Исходный текст - 212 байт.

Исполняемый код Primer.exe после трансляции - 539 байт

Место низкоуровневых языков программирования

- Для прямого программного доступа к аппаратным ресурсам системы
- Для создания минимальной по размеру и по времени исполнения программы
- Инструмент анализа исполняемых кодов при отсутствии их исходных текстов
- Уникальное учебное средство для понимания принципов работы вычислительной системы: работы процессора, его взаимодействия с памятью и аппаратурой

Организация работы

- Презентация лекции предоставляется в Учебных материалах MS Teams для предварительного ознакомления с темой лекции.
- Сама лекция расширяет тему презентации, сопровождает пояснениями и примерами, ответами на вопросы и т.д..
- Лекции и практические занятия - единое целое. Строгого деления не будет
- Вести конспект лекций и иметь тетрадь для работы на практических занятиях и лекциях (полезно иметь: карандаш + ластик)
- Для приобретения и закрепления навыков работы выдаются индивидуальные задания в рамках ЛР и самостоятельной домашней работы
- Последняя часть заданий выдаются в рамках КР

Критерии оценки успеваемости

- умение ответить на контрольные вопросы по каждой теме
- успешное выполнение контрольных работ по предлагаемым темам
- грамотное, самостоятельное выполнение и защита индивидуальных заданий
- успешная сдача экзамена

Помнить: умения и навыки по разработке и реализации алгоритмов на уровне команд процессора приобретаются постепенно, пропорционально затраченному на работу времени

Правила работы

- Работа на занятиях, ведение личного конспекта
- Выдаваемое задание (общее или индивидуальное) должно быть обязательно выполнено к указанному сроку. В процессе выполнения возможны любые онлайн консультации в Teams
- Выполненное задание присылается через чат в Teams в случае дистанционного режима учебы. Защита выполненной работы – на занятии. В защиту работы входят любые контрольные вопросы по теме и вопросы по выполнению задания.
- Следующее задание выдается только после успешной защиты предыдущего и при условии выполнения контрольной работы по предыдущей теме.
- Аттестация текущего контроля происходит 2 раза на основе результатов текущей работы студента.
- Студенты, не выполнившие в семестре программу курса и не показавших устойчивых навыков низкоуровневого программирования, аттестуются с оценкой «неуд». Будет предоставлена возможность двух переэкзаменовок в течение первого месяца следующего семестра.