Лектор: А.Д.Хапугин

Основы программного конструирования

Лекция 7. Инструменты программирования, начало Модулы-2

Материалы доступны в Интернете по адресу: http://www.excelsior.ru/afti/

Вопрос: сколько %% времени тратиться на написание кода, а сколько — на его чтение?
А усилий?

Задание: прочитайте программу:

Прочитайте вот эту программу из предыдущих лекций:

```
0 500
1 500 501 502
2 502 200
4 500 600
3 500
7 0
```

А теперь так:

0 500 (считать символ с УВВ и положить в ячейку 500)
1 500 501 502 (сравнить ячейки 500 и 501, результат сравнения занести в 502)
2 502 200 (взять результат сравнения из адреса 502, и если не тот, то перейти на 200)
4 500 600 (прибавить к слову по адресу 500 значение 600)
3 500 (увеличить слово по адресу 500 на 1)
7 0 (безусловно перейти на команду с адресом 0)

А так?

```
in      500
cmp     500 501 502
jc      502 200
add     500 600
inc     500
jmp     0
```

Вывод: удобнее обозначать команды на человеческом языке.

Еще сложности

При попытках исправить ошибки в программе изменяются адреса как команд так и данных.
Значит, после каждого перемещения данных нужно пройтись по всей программе, и исправить все места, где есть обращение к этим данным.
Представьте, как трудно найти такое место, если, например, адрес используется из регистра?

После добавления/убавления команд приходится пересчитывать длины переходов или их адреса.

В любой реальной программе сотни и тысячи команд!

Вывод: процесс нужно автоматизировать, написав инструментальные программы для программиста.

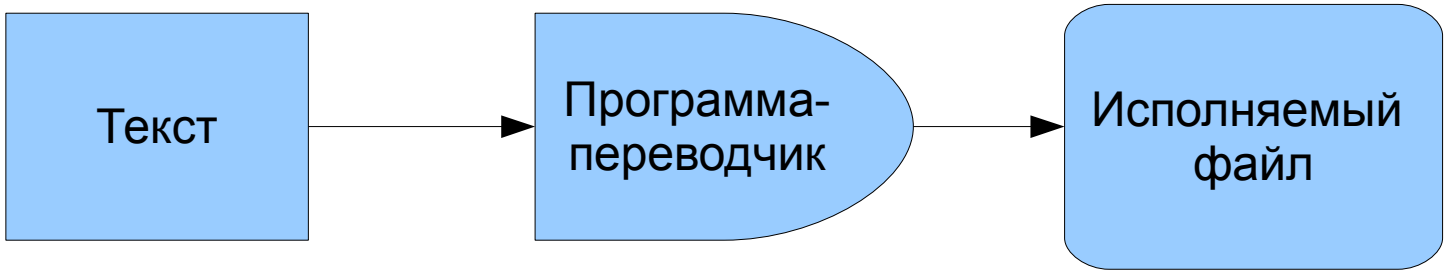
Ассемблер

```
symbol:  byte
counters: word(256)

input-symbol:
        in      177560,r0
        tst     r0
        jz      input-symbol
        mov     r0,$symbol
        ret

start:  call    input-symbol
        tst     $symbol
        jz      finish
        mov     r0, counters
        add     r0, $symbol
        inc     $r0
        jmp     start
finish: ret
```

Автоматизация



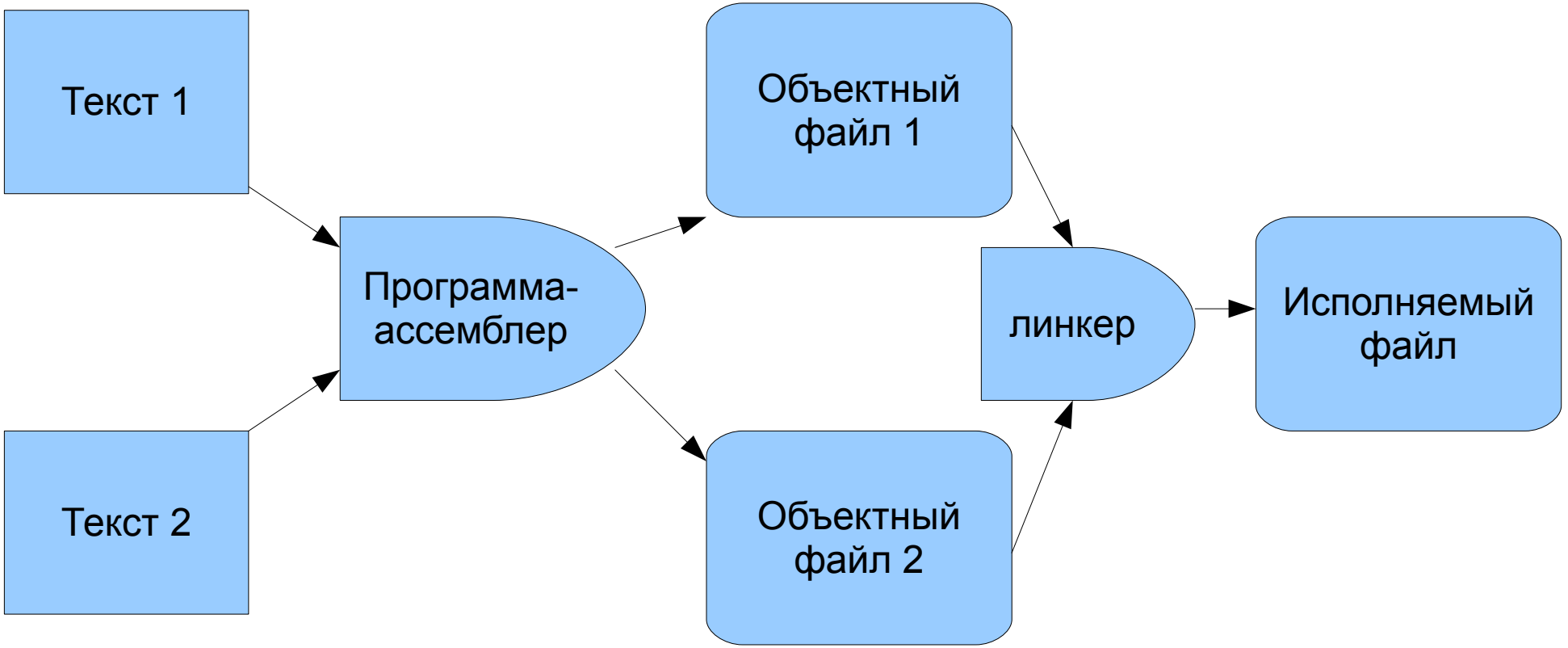
Ассемблер

Ассемблер = программа

Ограничения:
- мало памяти
- сложность разработки

Следствия:
- синтаксис (вид текста) выбирается так, чтобы еще был понятный но уже было легко разбирать
- текст должен быть устроен так, чтобы достаточно было его прочитать сверху вниз как можно меньше раз, и принять правильные решения для каждой из строк

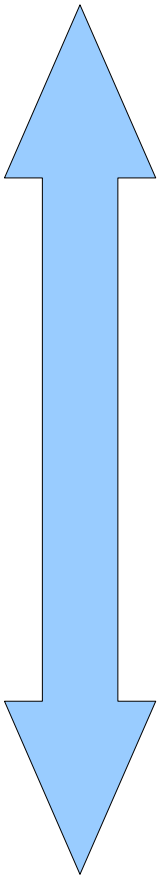
Ассемблер, линкер



Ассемблер: порождает промежуточное представление, программы, в котором есть информация о существовании объектов (переменных, процедур), но нет информации об их конкретном расположении.
Линкер: принимает решения о расположении объектов в памяти и настраивает все команды обращения к данному объекту на конкретный адрес. Порождает исполняемый формат.

Языки высокого уровня

Верх — то, как думает человек



Низ — то, как работает машина

Языки «высокого уровня» - заполняют еще одну ступень промежутка между машиной и человеком. Исторически ЯВУ пытаются создавать либо «от математики» (FORTRAN, PL-1, ALGL), либо «от компьютера» (B, C). Выживают те, которые позволяют решать реальные задачи эффективнее своих предшественников. Истина — где-то посредине.

Языки ассемблера: нижний уровень. Программы составляются из машинных, а не человеческих действий, но форма их представления уже удобнее для человека