Архитектура ЭВМ и язык ассемблера

Лисид Лаконский

February 2023

Содержание

1 Архитектура ЭВМ и язык ассемблера - 01.02.2023

Ассемблер – **машинно-ориентированный язык** и позволяет максимально использовать ресурсы процессора.
Ассемблер **тесно связан с архитектурой**.
Ассемблер – **мнемоническое отображение команд процессора**.

Стадии выполнения операции: **считывается код команды** — **считывается первый аргумент** — **считывается второй аргумент** — **выполнение операции** — **пересылка операции**

Этапы создания программы:

1. Создание текстового файла с мнемонической записью
2. Создание объектного модуля программы — результат трансляции без привязки к библиотечным модулям и адресам
3. Компоновка (линковка) — привязка модулей и адресов
4. Вывод загрузочного модуля

Во время компоновки может быть прикреплен **отладчик**
Отладчики бывают нескольких видов:

1. **Объектный** — проверяет только сам исполняемый файл
2. **Системный** — проверяет работу программы в системе

BIOS обеспечивает только ввод и вывод, операционная система — это другое
Макросы подставляются на этапе трансляции
Существует множество трансляторов языка ассемблера: MASM, NASM, TASM и другие

COM формат исполняемых файлов не имеет заголовка, его максимальный размер — 64 Кб
EXE улучшен относительно COM, не ограничен по размеру, может содержать несколько сегментов кода

1.1 Hello, world

1.1.1 TASM/NASM, MS DOS

```
section .text
    org 0x100

    mov ah, 0x9
    mov dx, hello
    int 0x21

    mov ax, 0x4c
    mov al, 0
    int 0x21

section .data
        hello DB "Hello, world!", 0xd, 0xa, '$'
```

1.1.2 NASM, UNIX

```
global _start

section .data
    ; String, which is just a collection of bytes, 0xA is newline
    str:     db 'Hello, world!',0xA
    strLen:  equ $-str

section .bss

section .text
_start:
    mov     edx, strLen     ; Arg three: the length of the string
    mov     ecx, str        ; Arg two: the address of the string
    mov     ebx, 1          ; Arg one: file descriptor, in this case stdout
    mov     eax, 4          ; Syscall number, in this case the write(2) syscall:
    int     0x80            ; Interrupt 0x80

    mov     ebx, 0          ; Arg one: the status
    mov     eax, 1          ; Syscall number:
    int     0x80
```