Синтаксис языка Ассемблера

Синтаксис языка Ассемблера

Ассемблер

программа, используемая для преобразования исходной программы на языке Ассемблера в машинный код

Язык Ассемблера

система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде

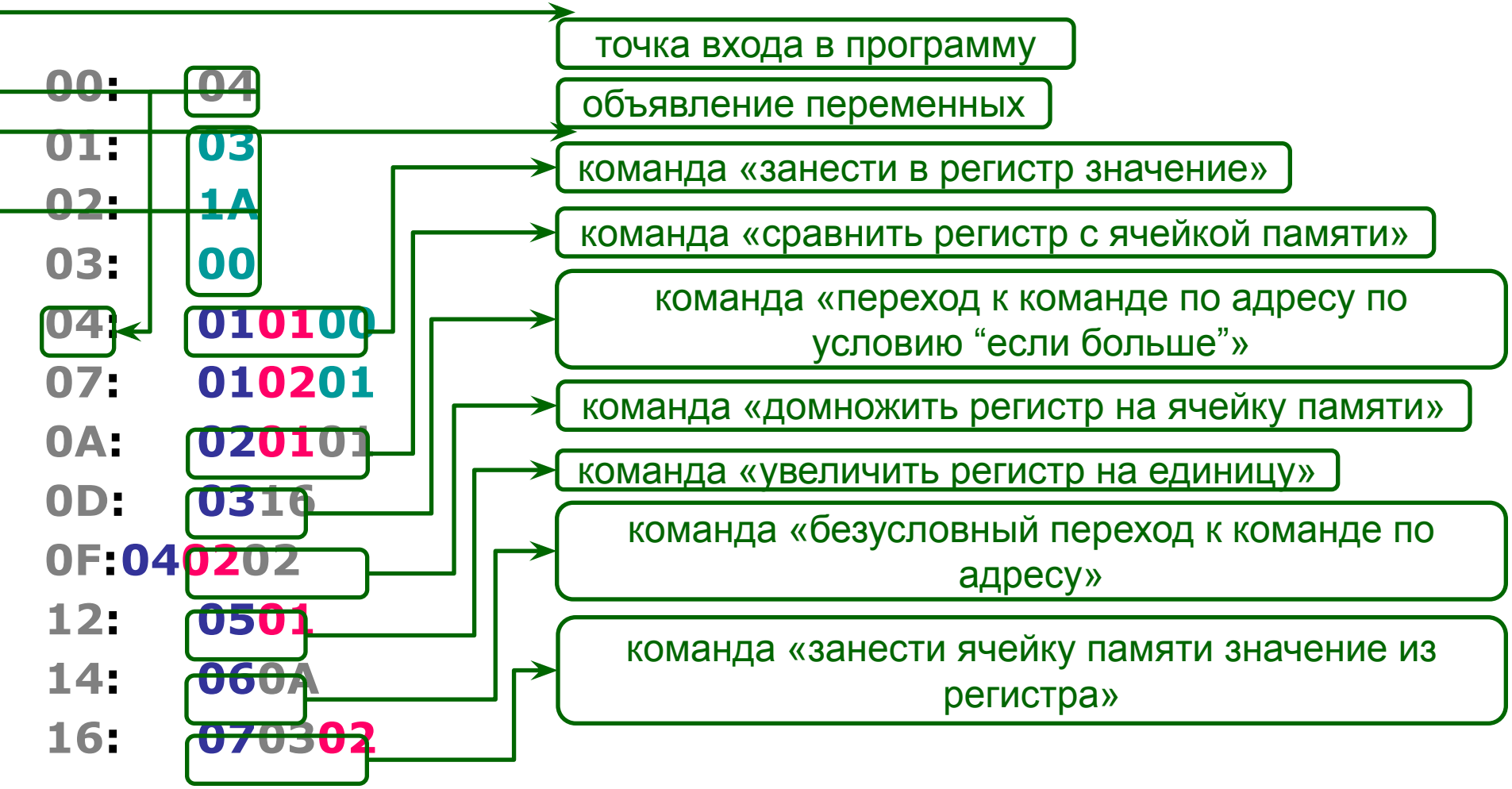
Синтаксис языка Ассемблера

Пример программы в машинном коде для некоторой архитектуры

```
0403 1A00 0101 0001
0201 0201 0103 1604
0202 0501 060A 0703
02
```

Синтаксис языка Ассемблера

Пример программы в машинном коде



Виды предложений языка Ассемблера

Инструкции

Макрокоманды

Директивы

Комментарии

Синтаксис языка Ассемблера

Структура программы на языке Ассемблера

```
.data
    ; входные данные
    x db 3
    y dw 26

.data?
    ; выходные данные
    z dd ?

.code

    start:
    mov AX, x
    add AX, y
    mov z, AX

end start
```

Синтаксис языка Ассемблера

Структура программы на языке Ассемблера

```
.data
    ; входные данные
    x db 3
    y dw 26

.data?
    ; выходные данные
    z dd ?

.code

    start:
    mov AX, x
    add AX, y
    mov z, AX

end start
```

Структура программы на языке Ассемблера

Программа на ассемблере представляет собой совокупность блоков, называемых *сегментами*.

Сегменты программы имеют назначение, соответствующее типам сегментов памяти (кода, данных и стека).

Каждый сегмент состоит из совокупности отдельных строк, называемых *предложениями языка Ассемблера*.

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

Система команд процесора

- .286
- .386
- .486

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

модель сегментации памяти и способ передачи параметров

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

сохранение регистра идентифи-каторов

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

подключение заголовочных файлов

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

определяет типы данных и константы, используемые операционной системой

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

прототипы функций по работе с пользовательским интерфейсом (окна, кнопки, стандартные диалоги)

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

прототипы функций ядра операционной системы (память, процессы, файлы)

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

gdi32.inc

прототипы функций интерфейса графических устройств

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

подключение библиотеки, в которой находится реализация нужных функций

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

сегмент инициализированных данных

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

.data?

сегмент неинициали-зированных данных

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title   db "Message",0
message db "Hello, World!",0
```

имя переменной

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

тип переменной

db – байт

dw – слово

dd – 2-е слово

dq – 4-е слово

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

значение переменной

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",
```

```
a db 105
b db 415
c dw 415
d dw 415
arr db 1,2,3,4,5
s1 db “abc”
s2 db “abc”,0
s3 db “a”,32,”c”,0
```

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

```
arr1 dw 1,2,5
  dup(0),6,7,3
  dup(9)
```

Простейшее приложение

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title     db "Message",0
message   db "Hello, World!",0
```

```
a db ?
b dw 7 dup(?)
d dd
e dq ? dup(?)
f db dup(?)
g dw 56
h dd 10 dup(4)
```

Простейшее приложение

```
.code

start:

    push MB_OK
    push offset title
    push offset message
    push 0
    call MessageBox

    push 0
    call ExitProcess

end start
```

Простейшее приложение

```
.code

start:

    push MB_OK
    push offset title
    push offset message
    push 0
    call MessageBox

    push 0
    call ExitProcess

end start
```

начало
сегмента кода

Простейшее приложение

```
.code

start:

    push MB_OK
    push offset title
    push offset message
    push 0
    call MessageBox

    push 0
    call ExitProcess

end start
```

конец программы

Простейшее приложение

```
.code

start:

    push MB_OK
    push offset title
    push offset message
    push 0
    call MessageBox

    push 0
    call ExitProcess

end start
```

точка входа в программу

Синтаксис языка Ассемблера

Формат инструкции или макрокоманды

[метка:] КОП [список операндов]

Формат директивы

[имя] директива [список операндов]

Синтаксис языка Ассемблера

Операнды

- обозначения регистров;
- числовые и текстовые константы;
- метки и имена переменных;
- знаки операций;
- зарезервированные слова.

Синтаксис языка Ассемблера

Машинные команды могут

– не иметь операндов

```
ret
```

– иметь один операнд

```
inc EAX
```

– иметь два операнда

```
add EBX, 1
```

Виды операндов

Регистровый операнд

обозначает регистр процессора (имя регистра)

Адресный операнд

обозначает адрес некоторой ячейки памяти

Непосредственный операнд

значение, которое указывается непосредственно в команде

Синтаксис языка Ассемблера

Для адресных операндов можно использовать различные *методы адресации*

Прямая адресация

в команде прямо указывается адрес (смещение) ячейки памяти

Косвенная адресация

в команде указываются регистр(ы), в которых находятся адрес (или часть адреса) ячейки памяти

Синтаксис языка Ассемблера

Полное выражение для вычисления адреса ячейки памяти при косвенной адресации:

регистр + масштаб * регистр + число

- **регистр** — базовый регистр
- **масштаб** — 1, 2, 4 или 8
- **регистр** — индексный регистр
- **число** — смещение

Синтаксис языка Ассемблера

Виды косвенной адресации определяются составом выражения для вычисления адреса, например:

Косвенная базовая

```
inc [EBX]
```

Косвенная базовая со смещением

```
inc [EBX + 10]
```

Косвенная базовая индексная со смещением

```
inc [EBX + 4*ESI + 10]
```

и т.д.

Синтаксис языка Ассемблера

Большинство машинных команд имеют два операнда, один из которых является ***источником***,

другой – ***приемником***.

Допустимы следующие сочетания операндов:

Приемник	Источник
Регистровый операнд	Регистровый операнд
Регистровый операнд	Адресный операнд
Адресный операнд	Регистровый операнд
Регистровый операнд	Непосредственный операнд
Адресный операнд	Непосредственный операнд