

Информатика

Занятие 2

Грозов Владимир Андреевич

va_groz@mail.ru

Simple Assembler

Simple CPU

| | | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|-----|----------|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|-------|
| | | 00 | 30 | 0a | 34 | 03 | 21 | 93 | 18 | 05 | 49 | 22 | 05 | ef | ef | 47 | 8c a0 |
| R0: | 00001010 | 01 | 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| R1: | 00000011 | 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| R2: | 10010011 | 03 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| R3: | 00000000 | 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| RC: | 00001000 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| RF: | 00000000 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 07 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 08 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 09 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 0a | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 0b | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 0c | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 0d | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 0e | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | 0f | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

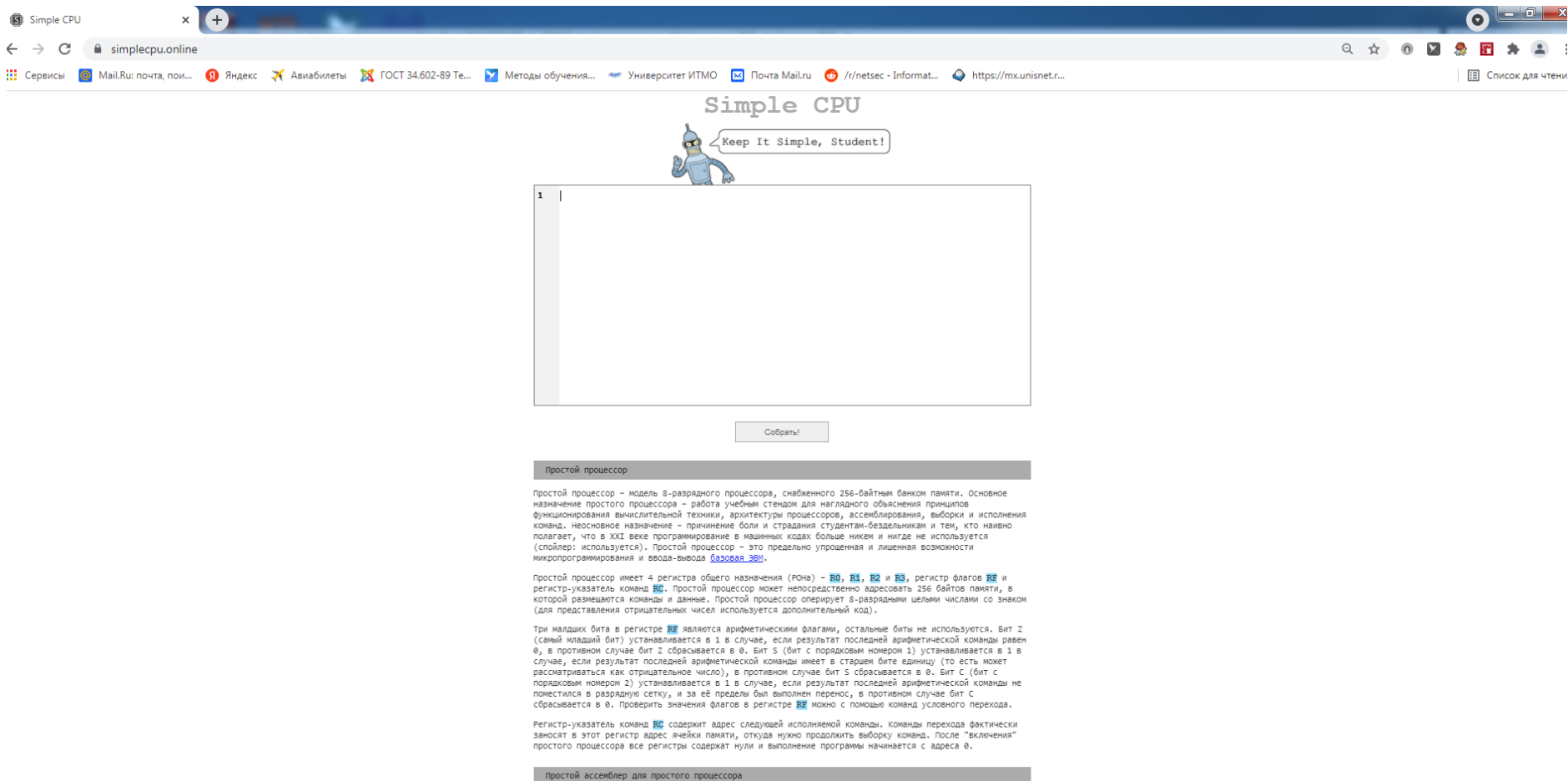
Load

Next

Reset

Simple Assembler

<https://simplecpu.online/>



Simple CPU

Keep It Simple, Student!

1 |

Собрать!

Простой процессор

Простой процессор – модель 8-разрядного процессора, снабженного 256-байтным банком памяти. Основное назначение простого процессора – работа учебным стендом для наглядного объяснения принципов функционирования вычислительной техники, архитектуры процессоров, ассемблирования, выборки и исполнения команд. Неосновное назначение – причинение боли и страдания студентам-бездельникам и тем, кто наивно полагает, что в XXI веке программирование в машинных кодах больше никем и нигде не используется (спойлер: используется). Простой процессор – это предельно упрощенная и лишенная возможности микропрограммирования и ввода-вывода [байтосвая 386](#).

Простой процессор имеет 4 регистра общего назначения (РОЧ) – [R0](#), [R1](#), [R2](#) и [R3](#), регистр флагов [R4](#) и регистр-указатель команд [R5](#). Простой процессор может непосредственно адресовать 256 байтов памяти, в которой размещаются команды и данные. Простой процессор оперирует 8-разрядными целыми числами со знаком (для представления отрицательных чисел используется дополнительный код).

Три младших бита в регистре [R4](#) являются арифметическими флагами, остальные биты не используются. Бит Z (самый младший бит) устанавливается в 1 в случае, если результат последней арифметической команды равен 0, в противном случае бит Z сбрасывается в 0. Бит S (бит с порядковым номером 1) устанавливается в 1 в случае, если результат последней арифметической команды имеет в старшем бите единицу (то есть может рассматриваться как отрицательное число), в противном случае бит S сбрасывается в 0. Бит C (бит с порядковым номером 2) устанавливается в 1 в случае, если результат последней арифметической команды не поместился в разрядную сетку, и за её пределы был выполнен перенос, в противном случае бит C сбрасывается в 0. Проверить значения флагов в регистре [R4](#) можно с помощью команд условного перехода.

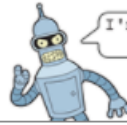
Регистр-указатель команд [R5](#) содержит адрес следующей исполняемой команды. Команды перехода фактически заносит в этот регистр адрес ячейки памяти, откуда нужно продолжить выборку команды. После "включения" простого процессора все регистры содержат нули и выполнение программы начинается с адреса 0.

Простой ассемблер для простого процессора

Simple Assembler

Simple CPU

<https://simplecpu.online/>



I'm gonna go build my own CPU!
With JTAG and buffers!

```
1 R00 <- @0xA0
2 R01 <- @0xA1
3 R00 <- R00 + R01
4 R02 <- @0xFF
5 R02 <- R02 - R00
6 RF <- R02 ~ R01
7 RC <- @LB1 (S)
8 @0xFF <- R02
9 LB1: @0xFF <- R00
10 RC <- @LB1
```

| | | | | | | | | |
|-----|---|---|---|---|---|---|---|---|
| R0: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R1: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R2: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RC: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RF: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 10 | a0 | 14 | a1 | 41 | 18 | f0 | 58 | 89 | a0 | 0d | 22 | ff | 20 | ff | c0 |
| 01 | 0d | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 03 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 07 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 08 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 09 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0a | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0b | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0c | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0d | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0e | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0f | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

Щаг!

Срагг!

Сбпол!

Simple Assembler. Характеристики

Характеристики

- 15 команд
- 256 байт памяти (матрица 16x16 однобайтовых ячеек)
- 4 регистра общего назначения
- 2 специальных регистра
- Данные в памяти записываются в виде шестнадцатеричных чисел
- Среда написания программ:
 - Блокнот/Wordpad/Notepad++

Работа с командной строкой

Запуск командной строки

ОС Windows: Меню «Пуск» → cmd

ОС Linux: Терминал

Важные команды

- **Переход на нужный диск**
- **Переход в нужный каталог (папку)**
 - cd путь_к_нужному_каталогу
 - Без русских букв!
- **Вызов справки**
 - help

Запуск виртуального микропроцессора

<https://simplecpu.online/>

R0, R1, R2, R3 – регистры
общего назначения

RC – регистр команд

RF – регистр флагов

Собрать! – загрузка
программы, компиляция и
последующая сборка

Шаг! – следующий шаг
выполнения программы

Старт! – запуск кода

Сброс! – начать работу
программы заново

| | | | | | | | | |
|-----|---|---|---|---|---|---|---|---|
| R0: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R1: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R2: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RC: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RF: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
| 00 | 10 | a0 | 14 | a1 | 41 | 18 | f0 | 58 | 89 | a0 | 0d | 22 | ff | 20 | ff | c0 |
| 01 | 0d | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 03 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 07 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 08 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 09 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0a | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0b | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0c | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0d | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0e | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0f | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

Шаг!

Старт!

Сброс!

Запуск виртуального микропроцессора

Регистры и оперативная память

Оперативная память – это общая память компьютера, которая продаётся в магазинах (платы).

Регистры команд – это процессорная память, располагается непосредственно в процессоре.

В языках высокого уровня (C, Pascal и т.д.) пользователь имеет доступ только к **оперативной памяти**.

В языке Assembler и в SASM пользователь имеет доступ также к **регистрам команд**.

Команды SASM

I. Работа с памятью:

1. **Запись данных из регистра в регистр (1 байт)**

R01 <- R00

2. **Запись данных из памяти в регистр (2 байта)**

R00 <- @0x11

R01 <- @111

3. **Запись данных из регистра в память (2 байта)**

@0x11 <- R00

@111 <- R01

4. **Запись числа в регистр (2 байта)**

R00 <- 10

R00 <- 0x10

Команды SASM

II. Арифметические операции

Арифметические операции в SASM можно выполнять только с регистрами!

1. Сложение (прибавление) (1 байт)

$R01 \leftarrow R01 + R00$

2. Вычитание (убавление) (1 байт)

$R03 \leftarrow R03 - R01$

3. Умножение (домножение) (1 байт)

$R02 \leftarrow R02 * R00$

4. Деление (1 байт)

$R00 \leftarrow R00 / R03$

Команды SASM

III. Сравнение и переходы:

Основаны на работе с флагами

1. Сравнение значений в регистрах (1 байт)

RF <- R01 ~ R02

2. Условный переход по флагу переноса (CF) (2 байта)

RC <- @0x12 (C)

RC <- @18 (C)

3. Условный переход по флагу знака (SF) (2 байта)

RC <- @0x23 (S)

RC <- @35 (S)

4. Условный переход по флагу нуля (ZF) (2 байта)

RC <- @0x0A (Z)

RC <- @10 (Z)

5. Безусловный переход (2 байта)

RC <- @0x31

RC <- @49

Команды SASM

III. Прочие команды

1. Занесение случайного числа в регистр (1 байт)

R01 <- ?

2. Инкремент значения регистра (1 байт)

R00 <- R00++

R01 <- R02++

Команды SASM. Метки

Программа без метки:

```
R00 <- 20
R01 <- 30
R02 <- 2
RF <- R00 ~ R01
RC <- @12(S)
R02 <- 100
R03 <- ?
R00 <- R00 + R02
@0xFF <- R00
```

Такая же программа с меткой:

```
R00 <- 20
R01 <- 30
R02 <- 2
RF <- R00 ~ R01
RC <- @Label(S)
R02 <- 100
R03 <- ?
Label: R00 <- R00 + R02
@0xFF <- R00
```

Работа в Simple CPU

Simple CPU



Keep It Simple, Student!

R00 <- @0xA0

R01 <- @0xA1

...

RC <- @LB1 (S)

...

Ненулевые значения в матрице — команды из скомпилированной программы.

Полужирным шрифтом в матрице выделены ячейки, которые соответствуют выполняемой на текущем шаге команде

```
1 R00 <- @0xA0
2 R01 <- @0xA1
3 R00 <- R00 + R01
4 R02 <- @0xF0
5 R02 <- R02 - R00
6 RF <- R02 ~ R01
7 RC <- @LB1 (S)
8 @0xFF <- R02
9 LB1: @0xFF <- R00
10 RC <- @LB1
```

| | | | | | | | | |
|-----|---|---|---|---|---|---|---|---|
| R0: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R1: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R2: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RC: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| RF: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 10 | a0 | 14 | a1 | 41 | 18 | f0 | 58 | 89 | a0 | 0d | 22 | ff | 20 | ff | c0 |
| 01 | 0d | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 03 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 07 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 08 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 09 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0a | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

Сравнение SASM, Assembler, Pascal, C

- Занесение числа в память:

| SASM | Assembler | Pascal | C |
|---------------------------|--|----------|---------|
| R00 <- 10 @0xA0 <- R00 | a1 resw 2 ... mov ax, 10 mov [a1], ax | a := 10; | a = 10; |

Сравнение SASM, Assembler, Pascal, C

- Арифметические операции:

| | SASM | Assembler | Pascal | C |
|--|---|---------------------------------------|-------------------------|--------------------|
| Прибавление одного числа к другому | R00 <- 10 R01 <- 5 R00 <- R00 + R01 @0xA0 <- R00 | mov ax, 10 mov cx, 5 add ax, cx | a := 10; a := a + 5; | a = 10; a += 5; |
| Вычитание одного числа из другого | R00 <- 10 R01 <- 5 R00 <- R00 - R01 @0xA0 <- R00 | mov ax, 10 mov cx, 5 sub ax, cx | a := 10; a := a - 5; | a = 10; a -= 5; |

Сравнение SASM, Assembler, Pascal, C

- Арифметические операции:

| | SASM | Assembler | Pascal | C |
|---|---|-----------------------------------|---------------------------|--------------------|
| Умножение одного числа на другое | R00 <- 10 R01 <- 5 R00 <- R00 * R01 @0xA0 <- R00 | mov ax, 10 mov bx, 5 mul bx | a := 10; a := a * 5; | a = 10; a *= 5; |
| Целочисленное деление одного числа на другое | R00 <- 10 R01 <- 5 R00 <- R00 / R01 @0xA0 <- R00 | mov ax, 10 mov bx, 5 div bx | a := 10; a := a div 5; | a = 10; a /= 5; |

В SASM арифметические операции можно осуществлять только с регистрами

Сравнение SASM, Assembler, Pascal, C

- Сравнение и передача управления:

| SASM | Assembler | Pascal | C |
|--|---|---|--|
| RF <- R00 ~ R01 RC <- @Label_1(S) ... RC <- @Label_2 Label_1: ... Label_2: | sub ax, bx JA Label_1 ... JMP Label_2 Label_1: ... Label_2: | if (a > b) then begin ... end; else begin ... end; | if a > b { ... } else { ... } |

SASM. Создание циклов

- Аналог цикла с предусловием (for или while):

```
R00 <- 0
R01 <- 10
R02 <- 1
LOOP: RF <- R00 ~ R01
      RC <- @END(Z)
      R00 <- R00++
      R02 <- R02++
      R02 <- R02++
      RC <- @LOOP
END: @0xFF <- R02
```

SASM. Создание циклов

- Аналог цикла с постусловием (do...while()):

```
R00 <- 0
R01 <- 10
R02 <- 1
LOOP: R00 <- R00++
      R02 <- R02++
      R02 <- R02++

      RF <- R00 ~ R01
      RC <- @END(Z)
      RC <- @LOOP
END: @0xFF <- R02
```