Информатика

Занятие 3

Грозов Владимир Андреевич
va_groz@mail.ru

Simple Assembler.
Выход за пределы байта

Положительные числа.

- R00 <- 250
- R01 <- 100
- R01 <- R01 + R00

- Чему будет равно значение, записанное в регистр R01?

Simple Assembler.
Выход за пределы байта

250 + 100 = 350.

НО:

$350_{10} = 101011110_2$

Когда в результате выполнения команды происходит выход за границы байта, та часть результата, которая оказалась за границами младшего байта, пропадает.
CF = 1.

350 - 256 = 94

Ответ: 0x5E (или 94)

Simple Assembler.
Выход за пределы байта

Отрицательные числа.

- R02 <- -150
- R03 <- -120
- R02 <- R02 + R03

- Чему будет равно значение, записанное в регистр R02?

Simple Assembler.
Выход за пределы байта

-150 + (-120) = -270.

НО:

$-270 = 1111111011110010_2$

Всё, что вышло за границы младшего байта, пропадает, CF = 1.

$11110010_2 = F2_{16} = 242_{10}$

Ответ: 0xF2 (или 242)

SASM. Битовое представление команд

Номер команды. 4 бита	**Куда (или 00). 2 бита**	**Откуда (или 00). 2 бита**	**Число / адрес ячейки**

Первый байт. Обязательный. **Второй байт. Есть не во всех командах**

Если в битовом представлении команды есть «?», то вместо него записывается 0.

SASM. Ошибки и предупреждения при компиляции

Ошибки

Строка x: тип ошибки (x – номер строки)

- **Неизвестная инструкция**
  Пример: R00 <- P00 + R01

- **‘ZZZ' выходит за пределы допустимого диапазона**
  Пример: R00 <- 256

- **Регистр назначения 'RXX' не совпадает с 'RYY'**
  Пример: R02 <- R01 + R02

Метки и переходы

Работа с метками

Метка – это более удобный вариант записи адреса!

- Внесение меток в код программы
- Компиляция программы
- Перекрёстные ссылки
- При выполнении программы происходит проверка меток и переход на соответствующие им адреса в памяти

SASM. Метки и переходы

```
R00 <- 0
R01 <- 10
LOOP: RF <- R00 ~ R01
RC <- @END (Z)
R02 <- @0xA0
…
RF <- R02 ~ R01
RC <- @Lb_if(S)
…
RC <- @Lb_endif
Lb_if:…
…
Lb_endif: …
…
RC <- @LOOP
END: @0xFF <- R02
```

SASM. Запись в разные ячейки 1

- **C:**

```
for (int i = 0; i < 10; ++i)
{
    a[i] = i;
}
```

- **SASM:**

Явного способа записать числа в разные ячейки (в цикле) не предусмотрено, но можно воспользоваться одной полезной хитростью.

SASM. Запись в разные ячейки 2

Порядок действий

1. Определить значения всех регистров, которые могут понадобиться в цикле (начальное значение счётчика итераций, количество итераций в цикле и т.д.)

2. Начало цикла – стандартное.
   RF <- R00 ~ R01
   RC <- @END (Z)

3. Основная команда, которая будет выполняться в цикле (например, если нам надо записать номера итераций цикла в строку @0xA0...@0xAF, то это будет @0xA0<-R00)

SASM. Запись в разные ячейки 3

4. Считаем, в каких ячейках будет храниться эта «основная» команда
5. Допустим, на данный момент получившийся фрагмент кода имеет следующий вид:

```
R00 <- 0
R01 <- 10
LOOP: RF <- R00 ~ R01
      RC <- @END (Z)
      @0xA0 <- R01
```

6. Команда @0xA0 <- R01 хранится в ячейках @7 и @8. @0xA0 – в ячейке @8.

SASM. Запись в разные ячейки 4

8. Пишем следующий фрагмент кода:
   R03 <- @8
   R03 <- R03++
   @8 <- R03

9. Таким образом можно по ходу работы программы менять адрес ячейки, с которой будет осуществляться работа

10. Далее – всё как и в обычном цикле

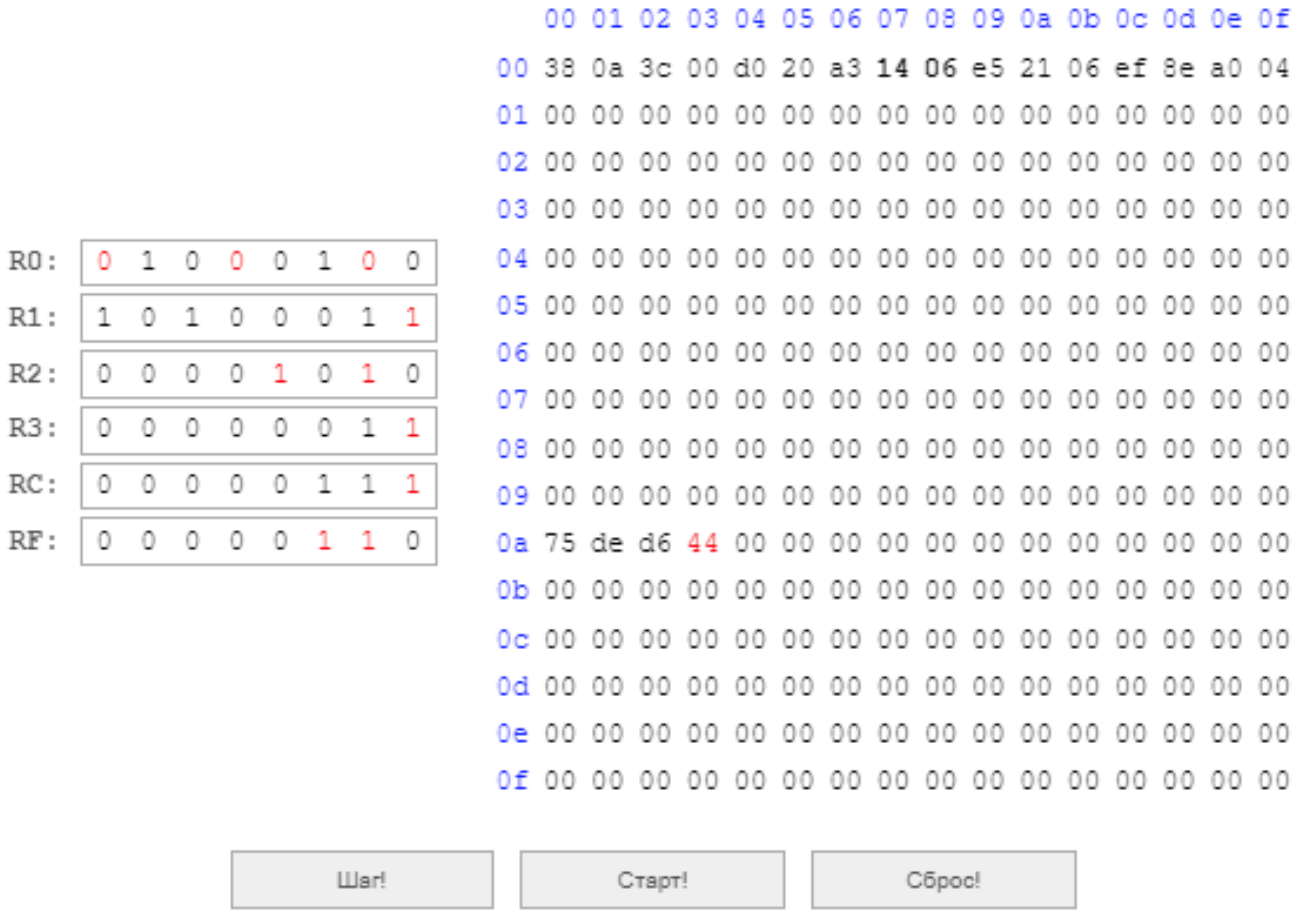
SASM. Запись в разные ячейки 4. Пример

Фрагмент кода. В ячейки @0xA0…@0xA10 записываются случайные числа

```
R02 <- 10
R03 <- 0
Loop:  R00 <- ?
            @0xA0 <- R00
            R01 <-@6
            R01 <- R01++
            @6 <- R01
            R03 <- R03++
            RF <- R03 ~ R02
            RC <- @Loop (S)
```

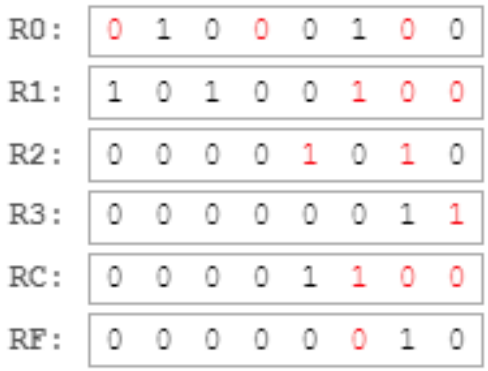
SASM. Запись в разные ячейки 5. Пример

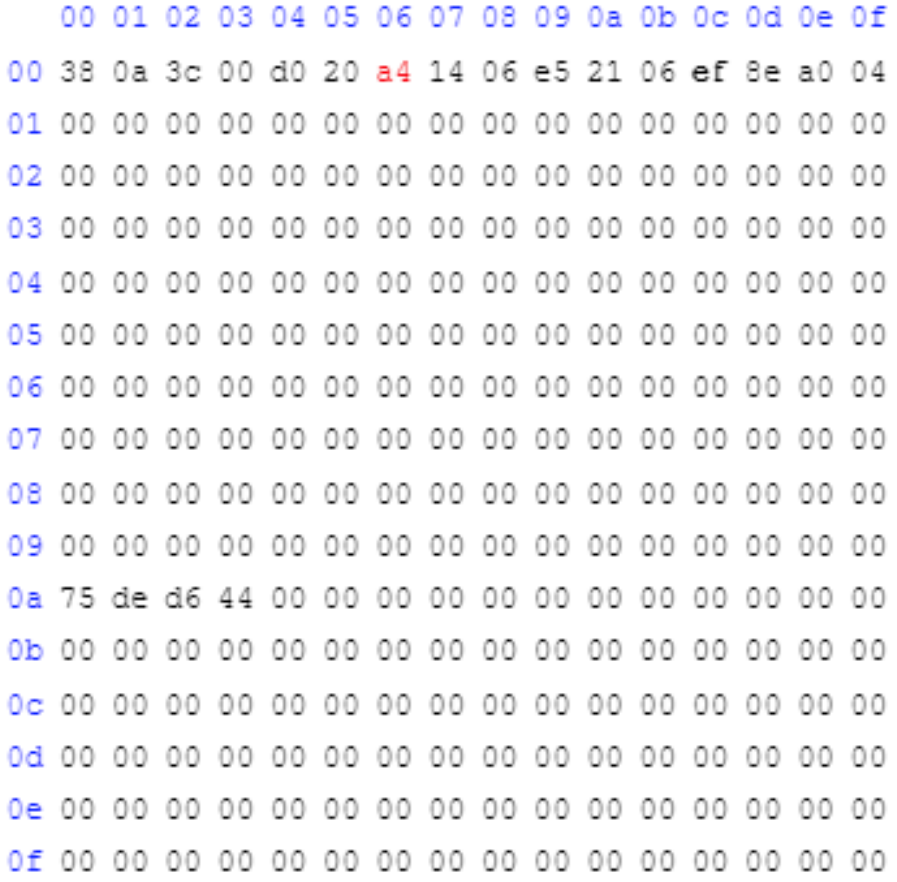
В очередную ячейку (@0xA3) записывается случайное число. Номер этой ячейки отображается в ячейке @0x06 (или @6)



SASM. Запись в разные ячейки 6. Пример

В цикле изменилось значение ячейки, в которую записывается случайное число (в ячейке @6 число 0xA3 изменилось на 0xA4)





SASM. Завершение программы

- Если не делать дополнительных действий, SCPU продолжит идти по ячейкам после того, как будет «пройден» весь код загруженной в него программы

- Часто это не страшно, но в некоторых случаях это может навредить, либо в какой-то момент может высветиться сообщение об ошибке.

- Чтобы этого избежать, надо в конец кода добавить строку следующего вида:

  FINISH: RC <- @FINISH

SASM. Ещё одна «хитрость» 1

- В SASM предусмотрено всего 4 регистра
- Для (условно) серьёзных программ этого может быть недостаточно
- RF и RC – специальные регистры, записывать в них числа нельзя!
- Можно использовать один и тот же регистр несколько раз и в нескольких целях

SASM. Ещё одна «хитрость» 2

Пример:

```
R00 <- 10
R01 <- 0
R02 <- 30
R03 <- 42
LOOP:  RF <- R01 ~ R00
       RC <- @END(Z)
       …
       @0xA0 <- R02    ;Старое значение R02
       R02 <- ?        ;Новое значение R02
       …
       R02 <- @0xA0    ;Возвращение старого значения R02
       …
       R01 <- R01++
       RC <- @LOOP
END: RC <- @END
```

SASM. Ещё одна «хитрость» 2

Таким образом, можно использовать одни и те же регистры для выполнения разных задач. С помощью этого подхода можно увеличить фактическое количество используемых переменных.

Но увлекаться этим тоже не нужно.