GNU Эмулятор данные и переходы

1. Безусловный переход.
2. Байты данных.
3. Организация условных переходов.

Безусловный переход

- Команда безусловного перехода позволяет передавать управление по определенному адресу, который представляет собой метку на которую передается управление.
- Формат команды:
  JMP label_name

```
JMP my_addr
  ;команды
my_addr:
  ;команда на которую выполняется переход
```

Безусловный переход

- Команда JMP используется при разработке программ, содержащих таблицу данных.
- Такая таблица данных создается с помощью директивы DB code
- Каждая директива определяет байт или последовательность байтов.

Байты кодов данных


```
;Пример байтов
DB “A”
DB “HELLO WORLD!”
```

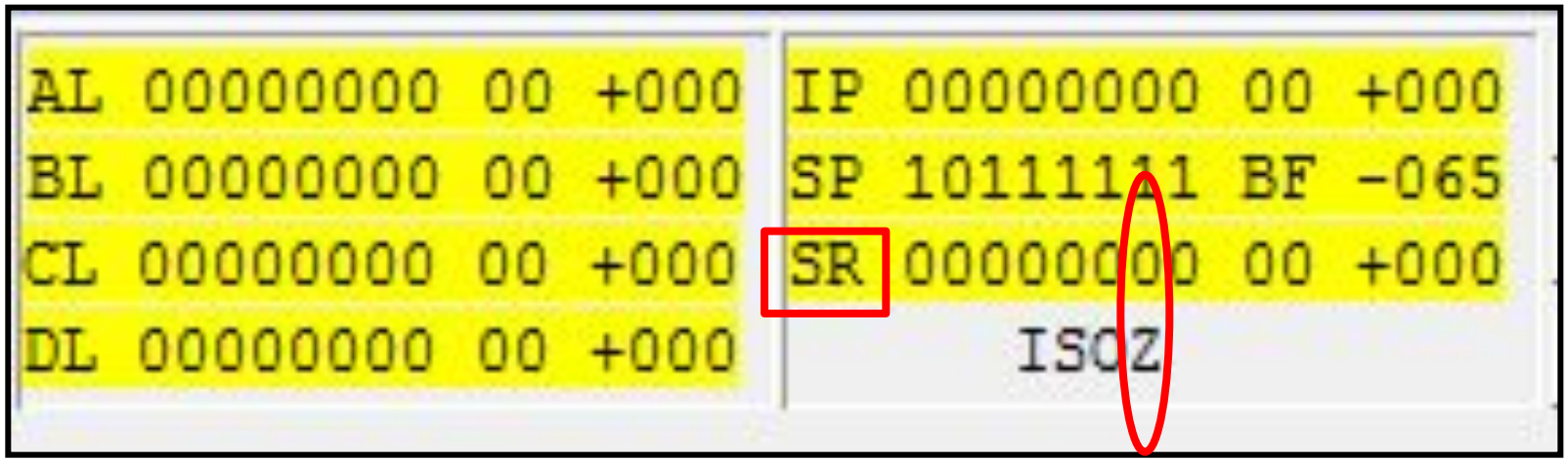
Source Code | List File | Configuration | T

```
JMP START ;Обход таблицы
  ;Таблица данных:
  DB "A"
  DB "!"
START:          ;Начало программы
  MOV AL,[02] ;Читать первый символ
  MOV [D5],AL  ;Писать в видеопамять
  MOV AL,[03]  ;Читать второй символ
  MOV [D6],AL  ;Писать в видеопамять
END
```

Упражнение №1, Откомпилируйте и выполните программу symbols.asm

Условный переход

- Процессор выполняет команду условного перехода в зависимости от значений битов регистра состояния SR.
- Для организации цикла можно использовать бит “Z” ,этот бит фиксирует состояние получения нуля (zero) при выполнении команды процессора (Z=1).



Компаратор

Сравнение кодов
команда CMP –компаратор.
Формат команды:
CMP A1,A2
Схема работы команды:
RES=A1-A2

Правила использования команды:
CMP R1,R2
CMP R,n
CMP R,[Addr]
Здесь R1,R2,R – регистры процессора,
Addr – адрес оперативной памяти.

Команды условного перехода

Для отслеживания состояния бита Z служат две команды ассемблера:
JZ my_label ;Z=1
JNZ my_label; Z=0

Упражнение №2. Откомпилируйте и выполните программу putsymbols.asm

Source Code | List File | Configuration | Tokens

```
JMP START   ; Перейти на начало
  DB "ABCDEFG" ;Строка из 7 символов.
START:
  MOV DL,C0 ;Начальный адрес видео памяти
  MOV AL,02 ;Адрес первого символа строки
  MOV CL,0  ;Счетчик цикла
GO:   ;Начало цикла
  MOV BL,[AL]
  MOV [DL],BL ;Символ в видеопамять
  INC CL   ;Новый шаг цикла
  CMP CL,7 ;Все шаги ?
  JZ STOP  ;Да, выйти
  INC DL   ;Нет, новый адрес видеопамяти
  INC AL   ;Адрес следующего символа
  JMP GO   ;В начало цикла
STOP:
END
```

Инкремент и декремент

- Для увеличения счетчика шагов в примере использована команда ассемблера INC R. Она увеличивает значение регистра R на единицу. В программе использован регистр CL в качестве счетчика. Можно использовать любой свободный регистр.
- Заметим, что процессор поддерживает также команду DEC R, которая уменьшает значение регистра R на единицу.

Цикл, команда DEC


```
MOV CL,n ;Число итераций цикла
go:
  ;Команды цикла
  DEC CL ;Уменьшить CL
  JNZ go ;Нет, бит Z=0
;Да, цикл закончен, бит Z=1
```

Задания

Контрольное задание 1. Создайте программу words.asm с таблицей данных из двух строк “TEST” и “OK!” .
Выведите эти строки. Так, как это показано на рисунке.

Задания

Контрольное задание 2. Создайте на основе программы putsymbols.asm программу putsymbols2.asm вывода строки в видеопамять, программируя цикл с использованием команд DEC и JNZ.

Задания

Контрольное задание 3. Создайте на основе программы putsymbols.asm программу putsymbols3.asm вывода строки в видеопамять.

- **Занесите число итераций – шагов цикла в регистр BL перед выполнением цикла.**
- **Внесите изменения в тело цикла, что бы команда CMP CL,BL выполнялась верно.**
- **Методическое указание: используйте стек для сохранения числа итераций.**

Задания

Контрольное задание 4.

- **На основе программы putsymbols.asm создайте программу putsymbols4.asm вывода строки в видеопамять так, что бы цикл перебора символов работал бы не по количеству символов, а по кодам символов.**
- **Выход из цикла выполняется после вывода символа «G».**

Задания

Контрольное задание 5. Создайте на основе программы putsymbols.asm программу putsymbols5.asm вывода строки в видео память, так что бы цикл перебора символов выполнялся бы по адресам.
Выход из цикла выполняется после получения адреса символа «G».