|  |  |
| --- | --- |
| 5.05.2016, 09:47     Ввести пять двухзначных чисел и вывести на экран самое большое из них | #**[4](http://www.cyberforum.ru/post9140526.html%22%20%5Ct%20%22_blank)** |
|

| **Assembler**[Выделить код](http://www.cyberforum.ru/) |
| --- |
|

|  |  |
| --- | --- |
| 12345678910111213141516171819 | org 100h**jmp** **start** max     **db**  0                   *; место под максимальное*array   **db**  12,25,64,10,87,0    *; массив*len     =   **$** - array           *; длина массива* **start**:       **mov**   **cx**,len             *; кол-во повторов*       **mov**   **si**,array           *; адрес массива*find:  **lodsw**                    *; берём 2 соседних элемента*       **cmp**   **ah**,**al**              *; сравниваем их друг-с-другом*       **jae**   next               *; переход, если первый больше*       **xchg**  **ah**,**al**              *; иначе: меняем их местами*next:  **dec**   **si**                 *; шаг назад*       **loop**  find               *; на повтор..*       **mov**   **[**max**]**,**ah**           *; отправляем макс. в переменную*        **ret**                      *; выход из программы* |

 |  |

 |

rd #10

wr 100                 - установили адрес обработчика прерываний от клавиатуры

EI                         - разрешили обработку прерывания процессором

rd #103

**out** 1                        - установили режим посимвольного ввода с клавиатуры

                                   **(**тогда прерывание будет формироваться после ввода каждого символа**)**

rd #11

**out** 1                     - установили флаг разрешения приема кодов в буфер клавиатуры

                                    и флаг разрешения прерывания от клавиатуры

M: **NOP**                 - пустой цикл – основная программа

**NOP**

**JMP** M

**in** 0                        - начало обработчика прерывания: ввод кода символа из буфера клав-ры

**OUT 1**                         - вывод кода символа на **OR**

**IRET**

**Учебная модель ЭВМ: контроллер дисплея**

Дисплей представляет собой модель внешнего устройства, реализующую функции символьного дисплея. Дисплей может отображать символы, задаваемые ASCII-кодами, поступающими на его регистр данных.



Дисплей включает:

* видеопамять объемом 128 слов (ОЗУ дисплея);
* символьный экран размером 8 строк по 16 символов в строке;
* четыре программно-доступных регистра:
	+ DR (адрес 0) — регистр данных;
	+ CR (адрес 1) — регистр управления содержит следующие флаги (устанавливаются и сбрасываются программно):
		- Е – флаг разрешения работы дисплея; при Е = 0 запись в регистры AR и DR блокируется;
		- А – флаг автоинкремента адреса; при А = 1 содержимое AR автоматически увеличивается на 1 после любого обращения к регистру DR – по записи или чтению.
	+ SR (адрес 2) – регистр состояния содержит единственный флаг *Err* – ошибки устанавливается аппаратно при попытке записать в регистр адреса число, большее 127, сбрасывается программно;
	+ AR (адрес 3) – регистр адреса.

Доступ в видеопамять осуществляется через «окно интерфейса»  необходимо сначала загрузить в регистр адреса AR номер ячейки видеопамяти, тогда обращение к регистру данных будет означать обращение (ввод или вывод) к ячейке видеопамяти с указанным адресом.

Контроллер дисплея выполняет две команды при выводе соответствующих кодов по адресу 1:

|  |  |
| --- | --- |
| **Десятичное число** | **Команда** |
| 101 | очистить дисплей (действие команды эквивалентно нажатию кнопки **Очистить** в окне **Дисплей**), при этом очищается видеопамять (в каждую ячейку записывается код пробела — 032), устанавливается в 000 регистр адреса AR и сбрасываются флаги ошибки Err и автоинкремента А; |
| 102 | сбросить флаг ошибки Err в регистре SR. |

Пример простой программы, выводящей на дисплей (в начало экрана) слово «Май»:

|  |  |  |
| --- | --- | --- |
| **Метка** | **Команда** | **Примечание** |
|   | RD #11 | ; включаем дисплей и устанавливаем |
|   | OUT 11 | ; флаг автоинкремента; |
|   | RD #0 | ; задаём начальный адрес |
|   | OUT 13 | ; выводимого слова; |
|   | RD #204 | ; ввод кода буквы «М» |
|   | OUT 10 | ; вывод на дисплей |
|   | RD #224 | ; ввод кода буквы «а» |
|   | OUT 10 | ; вывод на дисплей |
|   | RD #233 | ; ввод кода буквы «й» |
|   | OUT 10 | ; вывод на дисплей |
|   | HLT |   |

Ещё один пример программы, выставляющей в начало каждой строки её номер:

|  |  |  |
| --- | --- | --- |
| **Метка** | **Команда** | **Примечание** |
|   | RD #10 | ; включаем |
|   | OUT 11 | ; дисплей; |
|   | RD #0 | ; задаём начальный |
|   | WR R1 | ; адрес вывода; |
|   | RD #49 | ; вводим код |
|   | WR R2 | ; цифры «1»; |
|   | RD #8 | ; вводим число |
|   | WR R3 | ; повторений цикла; |
| **M1**:   | RD R1 | ; читаем текущий адрес |
|   | OUT 13 | ; и передаём в регистр адреса дисплея; |
|   | ADD #16 | ; увеличиваем адрес на 16 – на начало следующей строки; |
|   | WR R1 | ; сохраняем изменённый адрес; |
|   | RD R2 | ; читаем код цифры – номер строки; |
|   | OUT 10 | ; передаём код цифры на дисплей; |
|   | RD @R2+ | ; увеличиваем содержимое R2 (код цифры) на единицу; |
|   | JRNZ R3,**M1** | ; декремент R3 и переход на начало цикла, если R3 ≠ 0; |
|   | HLT |   |

## Учебная модель ЭВМ: подсистема прерываний

В модели предусмотрен механизм векторных прерываний. Источниками прерываний являются модели внешних устройств, в которых могут возникать события, требующие реакции программы.

Среди подключаемых ВУ формировать запросы на прерывания могут *контроллер клавиатуры* и *блок таймеров*, которым по умолчанию присваиваются вектора прерываний 0 и 2 соответственно. При этом вектор блока таймеров является общим для всех трёх таймеров, при необходимости обработчик прерывания может определить, какой из них вызвал прерывание по значению разрядов регистра SR состояния блока таймеров.

Таблица векторов прерываний располагается в ячейках ОЗУ 100 – 109, таким образом возможно обслуживать до 10 различных векторных источников прерываний. В ячейках таблицы должны располагаться адреса соответствующих программ – обработчиков прерываний. Эти адреса становятся известными после компиляции и заносятся в ячейки «вручную».

При вызове обработчика в стеке сохраняется адрес возврата (текущее значение РС) и значение регистр флагов IF, OV, S, Z. Возврат из обработчика прерывания осуществляется командой IRET, которая восстанавливает значения РС и регистра флагов.

Дисциплина обслуживания прерываний поддерживается с помощью программно-доступных флагов. Флаг IF разрешает/запрещает все прерывания, он устанавливается в «1» командой EI (*Разрешить прерывания*) и сбрасывается в «0» командой  DI (*Запретить прерывания*). После RESET процессора IF = 0, поэтому, если предполагается использовать прерывания, следует выполнить команду EI. При вызове любого обработчика IF аппаратно сбрасывается, поэтому если требуется реагировать на другие запросы внутри обработчика прерывания, в его начале следует выполнить команду EI. Команда IRET восстанавливает все флаги, в том числе IF = 1.

В системе команд модели имеются команды INT0 – INT9, которые позволяют программно вызывать обработчики прерываний независимо от значения IF.

На ВУ предусмотрены т.н. «локальные маски», позволяющие запретить/разрешить прерывание от любого источника независимо от остальных (см. описание соответствующих ВУ).

В качестве примера использования механизма прерываний рассмотрим вариант вывода кода нажатой клавиши на символьный дисплей, когда программа не проверяет готовность контроллера клавиатуры, а ждёт от него запроса на прерывание, может быть выполняя какие-либо другие полезные действия. В этом примере ожидание моделируется бесконечным пустым циклом.

|  |  |  |
| --- | --- | --- |
| **Метка** | **Команда** | **Примечание** |
|   | RD #11 | ; число 11 – в аккумулятор; |
|   | OUT 1 | ; включить клавиатуру и разрешить прерывание от неё; |
|   | OUT 11 | ; включить дисплей в режиме автоинкремента адреса; |
|   | RD #103 | ; передаём в контроллер код команды |
|   | OUT 1 | ; *Установить S в «1»* (режим посимвольного ввода); |
|   | EI | ; установить FI в «1» (разрешить прерывания); |
| **M1**:   | NOP | ; пустой |
|   | JMP **M1** | ; цикл; |
| **ОбрПр:** | IN 0 | ; считывание введённого символа из буфера в аккумулятор; |
|   | OUT 10 | ; передача ASCII-кода на символьный дисплей |
|   | IRET | ; возврат из прерывания. |

Внимание! Адрес обработчика прерывания от клавиатуры (значение метки ОбрПр) необходимо записать в ячейку 100, так как вектор клавиатуры по умолчанию равен 0. Если компилировать приведённую программу с нулевого адреса подряд, то адрес команды IN 0 (начало обработчика) равен 8. Поэтому число 8 следует перед запуском программы записать в ячейку 100.

## Учебная модель ЭВМ: специальные возможности

### Микрокомандный уровень

Реализация любой команды в модели учебной ЭВМ сводится к выполнению последовательности элементарных действий из довольно ограниченного множества микрокоманд (регистровые передачи, обращения к разным типам памяти и т.п.). Этот уровень моделирования обычно не виден пользователю при выполнении программы в пошаговом или автоматическом режиме.

Однако, если пользователь желает разобраться с последовательностью действий при выполнении командного цикла (надо помнить – абстрактной учебной ЭВМ), то он может подключить механизм *Микрокомандный уровень*. Для этого необходимо установить флажок *Режим микрокоманд* в меню *Работа* основного окна модели. На поведение модели при выполнении программы в автоматическом режиме этот флажок влияния не оказывает, а при работе в шаговом режиме по каждому нажатию кнопки *Шаг* выполняется только одна микрокоманда.

Для того, чтобы наблюдать за типом выполняемой микрокоманды, следует открыть окно *Микрокомандный уровень*(меню *Вид* команда *Микрокомандный уровень*. В этом окне отображается адрес и мнемокод выполняемой команды и список микрокоманд, которые её реализуют. С каждым нажатием кнопки *Шаг* выполняется очередная микрокоманда, указатель микрокоманд (стрелка) перемещается на следующую позицию, а результаты действия микрокоманды отображаются изменениями регистров процессора, РОН или ячеек памяти. После выполнения очередной микрокоманды допускается модификация регистров процессора и ячеек памяти.

Например, вот так выглядит окно *Микрокомандный уровень* в процессе выполнения команды WR @R1+
00 - содержимое PC передаётся в регистр адреса ОЗУ;
01 - чтение из ОЗУ;
02 - прочитанное слово передаётся в регистр команд CR;
03 - инкремент счётчика команд PC;
04 - 5-й разряд CR (адрес регистра) передаётся в регистр адреса РОН;
05 - чтение из РОН содержимого регистра (косвенного адреса);
06 - передача прочитанного содержимого регистра в промежуточный регистр адреса RA;
07 - передача адреса в регистр адреса ОЗУ;
08 - содержимое аккумулятора передаётся в регистр данных ОЗУ;
09 - запись в ОЗУ;
10 - инкремент содержимого регистра.

Для отключения режима *Микрокомандный уровень* не достаточно закрыть одноимённое окно - следует снять флажок *Режим микрокоманд* в меню *Работа*.

### Кэш-память

В модели реализован механизм полностью ассоциативной кэш-памяти (любая ячейка кэш замещает любую ячейку ОЗУ)

 

При подключении кэш-памяти можно выбрать её параметры:

* размер - 4, 8, 16 или 32 ячейки;
* режим записи - *сквозная* или *обратная* (при кэш-попадании сквозная запись осуществляется в ячейки кэш и ОЗУ, а при обратной - только в ячейку кэш и копируется в ОЗУ только при очистке ячейки);
* алгоритм замещения - *Случайный выбор*, *Очередь*, *С учётом бита использования*, причём эти алгоритмы могут выбирать из всех ячеек кэш или только из тех, в которые не производилось записи - если установлен флажок *С учётом бита записи*.

Ячейка кэш-памяти содержит три поля: тэг (признак), роль которого играет адрес ОЗУ, данные (содержимое ячейки ОЗУ) и признаки.

* Признак занятости **Z** отмечает занятые ячейки и устанавливается в «1», когда содержимое ячейки ОЗУ копируется в ячейку КЭШ. Сбрасывается при очистке кэш.
* Признак использования **U** устанавливается в «1» при каждом обращении к ячейке кэш. В тот момент, когда признаки **U** всех ячеек станут равными «1», все они сбрасываются в «0». Таким образом, признак **U** разделяет ячейки кэш на два непересекающихся подмножества - те, к которым обращались после последнего сброса (для них **U**=1) и те, к которым не обращались. Некоторые алгоритмы замещения используют эту информацию для выбора освобождаемой ячейки кэш.
* Признак записи в ячейку **W** устанавливается в «1» при записи в ячейку кэш. Алгоритм замещения может учитывать значения флагов при выборе освобождаемой ячейки кэш.

В процессе работы модели с подключённой кэш-памятью при обращении программы в ОЗУ осуществляется поиск требуемого адреса в кэш-памяти. В случае кэш-попадания обращение производится в ячейку кэш, при кэш-промахе обращение производится в ОЗУ и содержимое ячейки ОЗУ копируется в кэш.

При выполнении программы подсчитывается общее число обращений к ОЗУ, количество кэш-попаданий и количество обращений по записи.

Конечно, при работе программной модели ЭВМ с подключённой кэш-памятью нельзя наблюдать уменьшения времени решения задачи, но собираемая статистика позволит оценить эффективность применения кэш-памяти при использовании разных алгоритмов замещения.