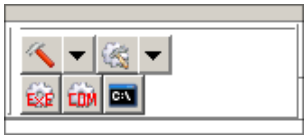
TutorForWindows

Открываем ASMTool, Файл->Новый, пишем программу, после сохраняем и компилируем и запускаем через шестеренку с молотком(вторая сверху):



Определение сегментов:

```
.8086
.model large        ; Используется модель памяти large (для сегментов кода и данных)
.stack 100h         ; Выделяем стек размером 256 байт (100h)
.data
  A dw 1, -3, 512, 0, 6, 2, -1024, '$'  ; Массив 16-битных чисел, завершаемый
символом '$'
  buf db 20 dup(0)    ; Новый массив, инициализированный нулями (20 байт)
.code
```

- `A` — массив целых чисел, который включает положительные и отрицательные значения, а также символ `$`, который обозначает конец массива.
- `buf` — буфер на 20 байт, не используется в текущей версии программы, но может быть использован для хранения строк.

Макрос для печати числа:

```
print_num MACRO reg
  mov bx, 10          ; Устанавливаем делитель 10 для деления числа на десятичные
цифры
  xor cx, cx          ; Очищаем CX, который будет использоваться как счетчик цифр
conv_loop_&reg:       ; Метка начала цикла преобразования
  xor dx, dx          ; Очищаем старшую часть регистра DX
  div bx              ; Делим AX на 10, результат в AX, остаток (цифра) в DX
  add dl, '0'         ; Преобразуем цифру в ASCII-символ
  push dx             ; Сохраняем цифру в стек
  inc cx              ; Увеличиваем счетчик цифр
  cmp ax, 0           ; Если AX не равен нулю, продолжаем деление
  jnz conv_loop_&reg
print_digit_&reg:     ; Метка начала цикла вывода цифр
  pop dx              ; Извлекаем цифру из стека
  mov ah, 2           ; Функция DOS для вывода символа
  int 21h             ; Вызов DOS прерывания для вывода символа
```

```
  loop print_digit_&reg  ; Цикл продолжается, пока CX не равен нулю
  mov dl, ' '        ; Выводим пробел после числа
  mov ah, 2          ; Функция DOS для вывода символа
  int 21h            ; Вызов прерывания DOS для вывода пробела
ENDM
```

- Этот макрос **`print_num`** выполняет преобразование числа, находящегося в `AX`, в строку и выводит его на экран по одной цифре.
  - Число делится на 10, остатки от деления (цифры) сохраняются в стеке.
  - Затем цифры извлекаются из стека и выводятся на экран с помощью прерывания **`int 21h`**.
  - После числа выводится пробел для разделения значений.

Основная программа:

```
start:
  mov ax, @data      ; Загружаем сегмент данных в AX
  mov ds, ax         ; Устанавливаем сегмент данных DS
  lea si, A          ; Загружаем адрес массива A в регистр SI
next_num:
  mov ax, [si]       ; Загружаем текущее значение из массива A в AX
  cmp ax, '$'        ; Проверяем, достигли ли символа конца массива ('$')
  je end_prog        ; Если да, завершаем программу
  cmp ax, 0          ; Проверяем, положительное ли число
  jge pos_num        ; Если больше или равно 0, обрабатываем как положительное
  mov bx, ax         ; Если число отрицательное, сохраняем его в BX
  mov dl, '-'        ; Загружаем символ '-' в DL для вывода
  mov ah, 2          ; Функция DOS для вывода символа
  int 21h            ; Вызов прерывания DOS для вывода символа '-'
  neg bx             ; Инвертируем число, чтобы сделать его положительным
  mov ax, bx         ; Переносим значение обратно в AX
pos_num:
  print_num ax       ; Вызываем макрос для вывода числа
  add si, 2          ; Переходим к следующему элементу массива
  jmp next_num       ; Переходим на следующую итерацию
end_prog:
  mov ah, 4Ch        ; Завершение программы через DOS
  int 21h            ; Вызов прерывания DOS для завершения программы
end start
```

- **`mov ax, @data`** и **`mov ds, ax`**: Инициализирует сегмент данных.
- **`lea si, A`**: Загружает адрес массива `A` в `SI`, который будет использован для перебора элементов массива.
- Цикл копирования чисел:
  - **`mov ax, [si]`**: Загружает текущее значение из массива `A` в регистр `AX`.

- `cmp ax, '$'`: Если встречен символ `$`, программа завершает свою работу.
- **Обработка отрицательных чисел:** Если число отрицательное (меньше нуля), оно инвертируется с помощью `neg`, и перед выводом числа выводится знак минус.
- `print_num ax`: Вывод числа с использованием макроса.
- `add si, 2`: Переход к следующему 16-битному числу в массиве.