Инструкция по работе с MASM Language VSCode Support

Шаг 1. Скачайте и установите Visual Studio Code

1. Перейдите на официальный сайт Visual Studio Code.
2. Скачайте Visual Studio Code.
3. Установите Visual Studio Code, следуя инструкциям установщика.

Шаг 2. Установите расширение MASM Language VSCode Support

1. Запустите Visual Studio Code.
2. Нажмите на иконку **Extensions** в левом боковом меню или используйте сочетание клавиш `Ctrl+Shift+X`.
3. В строке поиска введите `MASM Language VSCode Support`.
4. Найдите расширение от Gregory Ginzburg и нажмите **Install** или перейдите по ссылке и установите расширение.

Шаг 3. Распакуйте архив с примерами и настройками

1. Скачайте архив `masm_example_folder.zip` с сайта arch32.cs.msu.ru/semestr2/, содержащий пакет для работы с MASM, и настройками.
2. Распакуйте архив в удобное для вас место, не содержащее русских букв в пути. Например, в папку `C:\masm_example_folder`.
3. Откройте папку в VSCode:
   - В меню выберите **File > Open Folder**.
   - Найдите папку, куда вы распаковали архив, и нажмите **Select Folder**.

Примечание

Если у вас уже установлен пакет для **MASM**, то скачивать архив не нужно, достаточно указать путь к нему в файле `./vscode/settings.json`. Для этого добавьте в этот файл следующие строки:

```
{
    "masm.compilerPath": "YOUR PATH",
    "masm.linkerPath": "YOUR PATH",
    "masm.includePaths": ["YOUR PATH 1", "YOUR PATH 2"],
    "masm.libPaths": ["YOUR PATH 1", "YOUR PATH 2"]
}
```

Шаг 4. Как создавать новые файлы в VSCode

1. Убедитесь, что вы находитесь в нужной папке в VSCode.
2. Нажмите на иконку **Explorer** в левом боковом меню или используйте сочетание клавиш `Ctrl+Shift+E`.
3. В левой части окна найдите дерево файлов. Нажмите правой кнопкой мыши в пустом месте в **Explorer** и выберите **New File**.
4. Назовите новый файл, например, `example.asm`.
5. Файл откроется в редакторе. Теперь вы можете начинать писать код на MASM.

Шаг 5. Компиляция и запуск программ

Компиляция и запуск одного файла

1. Откройте любой из примеров или создайте новый файл `.asm`.
2. Чтобы запустить программу:
   - Нажмите на зеленую кнопку "треугольника" справа вверху (или используйте сочетание клавиш `Ctrl+F5`).
3. Чтобы запустить программу с дебаггингом:
   - Нажмите на иконку справа "жучка" справа вверху (или используйте сочетание клавиш `F5`).

Компиляция и запуск многомодульных программ

Если ваша программа состоит из нескольких `.asm` файлов, вам нужно настроить файл `tasks.json`, который отвечает за сборку проекта.

1. Откройте файл `tasks.json` в папке `.vscode`.
2. В параметре `files` укажите все файлы, которые нужно скомпилировать:

```
"files": [
  "main.asm",
```

```
  "module1.asm",
  "module2.asm"
]
```

Можно указывать относительные пути, тогда пути берутся от открытой в VSCode папки проекта:

```
"files": [
  "src/main.asm",
  "src/modules/module1.asm",
  "src/modules/module2.asm"
]
```

3. В параметре `output` задайте путь (абсолютный или относительный) к итоговому `.exe` файлу:

```
"output": "program.exe"
```

Вы также можете использовать переменные среды VSCode, например:

- `${file}` — полный путь к текущему открытому файлу.
- `${fileDirname}` — путь к папке текущего открытого файла.
- `${fileBasenameNoExtension}` — имя файла без расширения.
  Пример настройки для сборки многомодульной программы:

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "masmbuild",
      "label": "Build",
      "files": [
        "src/main.asm",
        "src/modules/module1.asm",
        "src/modules/module2.asm"
      ],
      "output": "program.exe",
      "compilerArgs": [
        "/c",
        "/coff",
        "/Zi",
        "/Fl",
        "/W3"
```

```
      ],
      "linkerArgs": [
        "/SUBSYSTEM:CONSOLE",
        "/DEBUG",
        "/MACHINE:X86"
      ]
    }
  ]
}
```

4. Сохраните файл `tasks.json`.
5. Чтобы запустить программу:
    - Нажмите на зеленую кнопку "треугольника" справа вверху (или используйте сочетание клавиш `Ctrl+F5`).
6. Чтобы запустить программу с дебаггингом:
    - Нажмите на иконку справа "жучка" справа вверху (или используйте сочетание клавиш `F5`).

Шаг 6. Отладка программ в Visual Studio Code

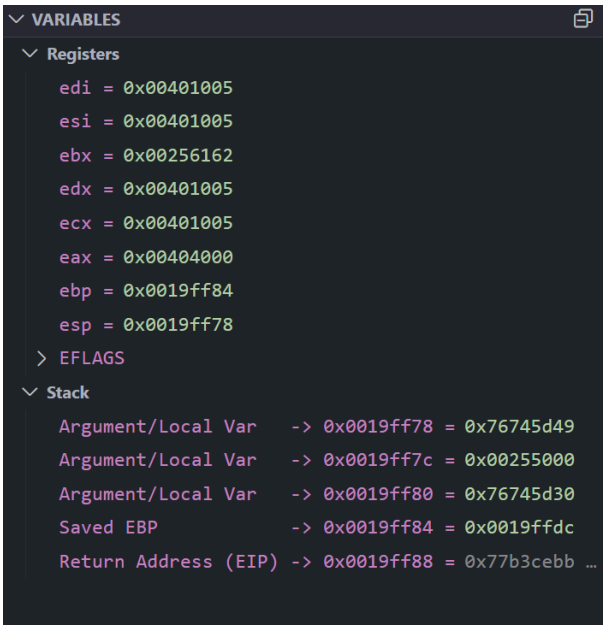
Запуск отладчика

1. Убедитесь, что у вас стоит хотя бы одна точка останова (breakpoint) в коде
    - Как добавить точку останова:
        - Найдите строку кода, куда хотите поставить точку останова.
        - Щёлкните левой кнопкой мыши слева от номера строки.
        - Появится красный кружок, указывающий, что точка останова установлена.
    - Управление точками останова:
        - Чтобы удалить точку останова, повторно щёлкните на красный кружок.
2. Нажмите на иконку справа "жучка" справа вверху (или используйте сочетание клавиш `F5`).

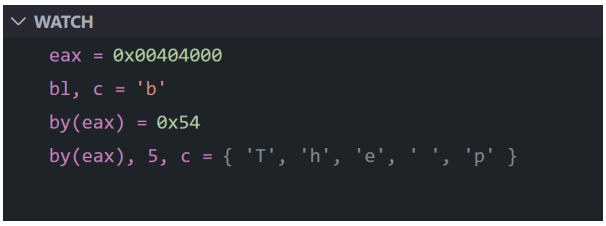
Интерфейс окна отладки

При запуске отладчика откроется несколько окон:

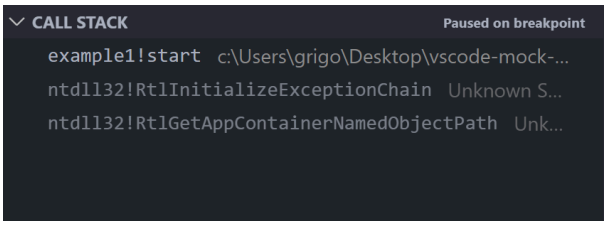
1. **Variables** (Переменные):
    - Показывает текущие значения основных регистров, флагов и стека.
    - Позволяет отслеживать изменения значений в реальном времени.



2. **Watch** (Наблюдение):
   - Сюда можно добавлять выражения, которые вы хотите отслеживать.
   - Например:
     - `eax` (значение регистра).
     - `bl, c` (значение регистра в формате `char`).
     - `by(eax)` (значение байта по адресу в `eax`).
     - `by(eax), 5, c` (значение 5 байтов (массив) по адресу в `eax` в формате `char`).
     - Подробнее о возможных выражениях можно прочитать ниже.
   - Для добавления нового выражения:
     - Щёлкните правой кнопкой мыши и выберите **Add Expression**.
     - Введите выражение и нажмите `Enter`.



3. **Call Stack** (Стек вызовов):
   - Отображает текущий стек вызовов функций.
   - Используется для отслеживания цепочки вызовов в программе.



4. **Debugger toolbar** (Меню для управления отладкой):

- Continue - Продолжить выполнение программы.
- Step Over — Выполнить текущую строку кода, не заходя внутрь функций (сочетание клавиш `F10`).
- Step Into — Перейти внутрь функции (сочетание клавиш `F11`).
- Step Out — Завершить выполнение текущей функции и вернуться к вызвавшему коду (сочетание клавиш `Shift+F11`).
- Stop — Остановить выполнение программы (сочетание клавиш `Shift+F5`).
- 

5. **Main window** (Окно с кодом программы)

- Отображает строчку, которая будет выполнена следующей.
- При наведении на переменную - отображается адрес переменной и ее значение.
- При наведении на регистр - отображается его значение.
- 

Использование выражений в Watch

Окно даёт возможность отслеживать значение регистров, переменных и более сложных выражений во время работы программы. Значением имени переменной (как и любого адресного выражения) является 32-битный адрес. Для того, чтобы увидеть содержимое соответствующих ячеек памяти перед адресным выражением нужно указать тип (`by` – байт, `wo` – слово, `dwo` – двойное слово), при этом выражение берут в скобки. Соответствующее значение по умолчанию печатается в виде 16-ричного числа; но можно указать формат печати (`b` – двоичное число; `d`, `u` – десятичное число со знаком или без знака; `h` – 16-ричное число; `c` – символ). Если нужно посмотреть несколько ячеек (например, элементы массива), нужно указать их количество.

Замечание: если указан тип, то любое выражение в скобках трактуется как адрес.

Синтаксис

- *watch-выражение* ::= *выражение* [*формат*] [, *количество*]
  | *тип*(*выражение*) [*формат*] [, *количество*]
- *тип* ::= `by` | `wo` | `dwo`
- *выражение* ::= *регистр* | *адресное выражение* | *константное выражение*
- *формат* ::= `b` | `d` | `u` | `h` | `c`

Примечание
Если имя переменной `x` совпадает с именем модуля `x` (если запущен `x.exe`), то ввод `x` в Watch приведёт к отображению адреса начала модуля `x`. Чтобы получить адрес переменной `x`, нужно ввести `x!x`.

Теперь вы готовы к разработке на MASM в Visual Studio Code. Удачи! 🎉