

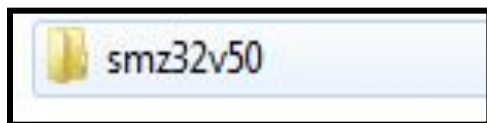
# **GNU Эмулятор**

## **КОМПИЛЯЦИЯ ПРОГРАММ**

- 1. Интерфейс эмулятора.**
- 2. Видео память.**
- 3. Оперативная память.**
- 4. Стек программы.**

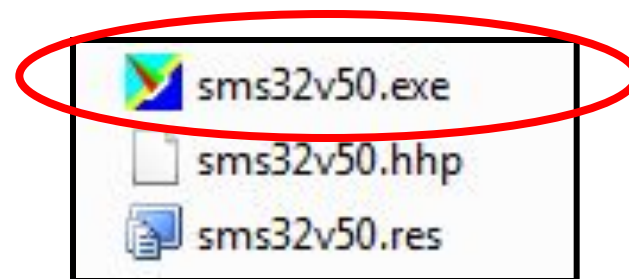
# Запуск эмулятора

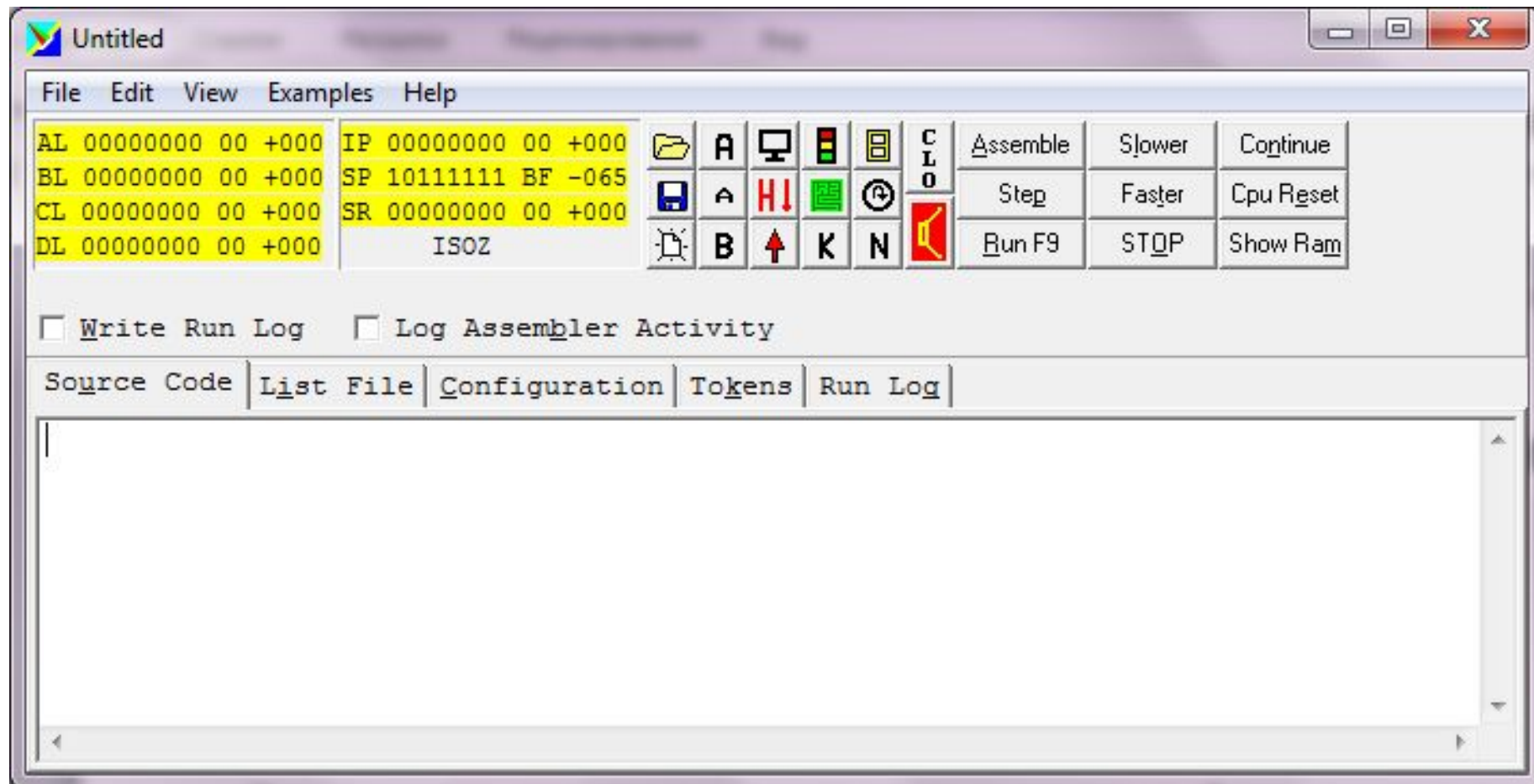
**GNU эмулятор Neil Bauers  
Microprocessor Simulator ver 5.0.**



**1. Используя проводник MS Windows найдите и откройте папку программы эмулятора.**

**2. Среди списка файлов программы найдите загрузочный файл программы, отмеченный пиктограммой и выполните его запуск.**



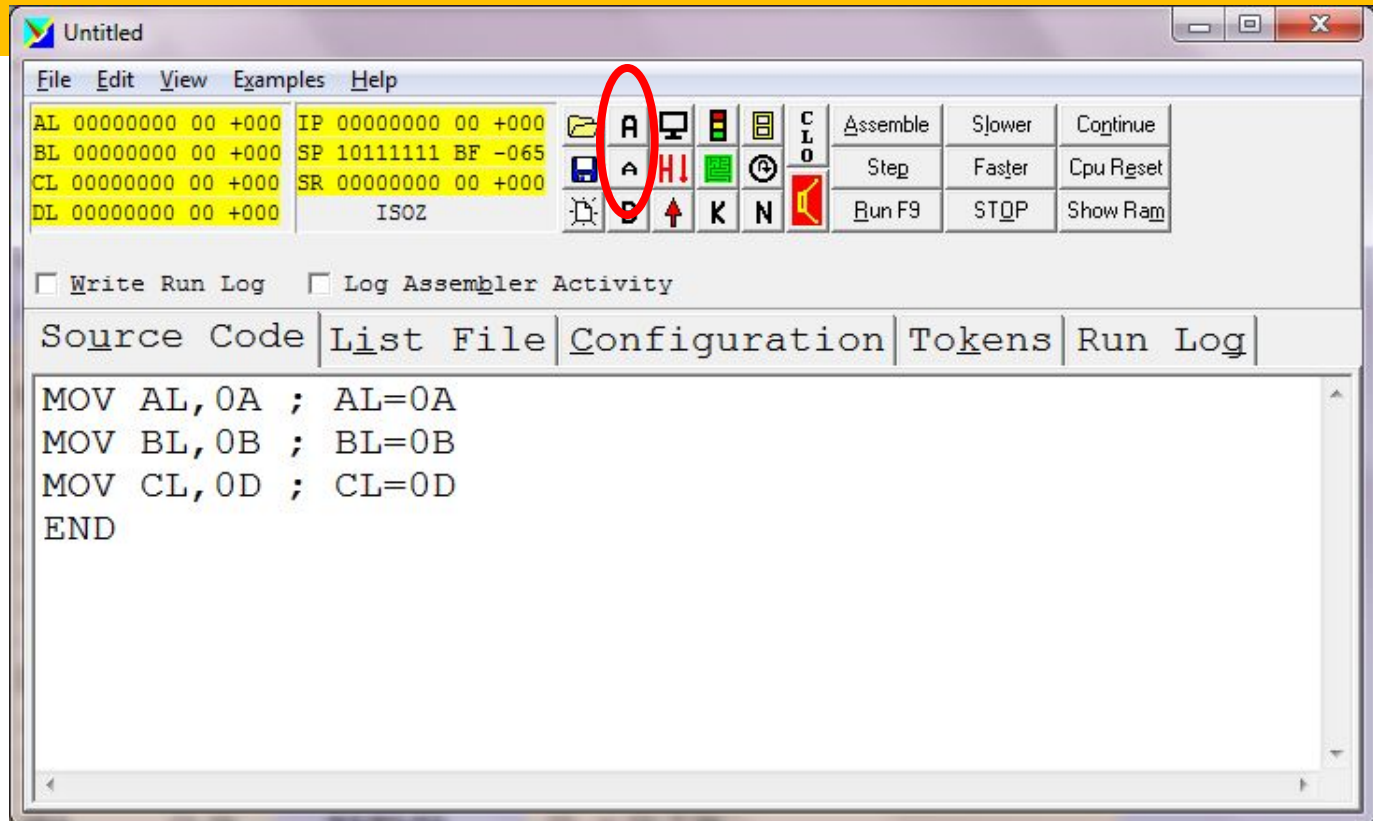


- ❑ Программа эмулятора моделирует работу восьмиразрядного микропроцессора.
  - ❑ Микропроцессор содержит четыре регистра общего назначения: AL, BL, CL, DL.
- 
- ❑ Регистр IP (Instruction Pointer) для адресации команд.
  - ❑ Регистр SP (Stack Pointer) для работы со стеком и регистр состояния SR (Status Register).
  - ❑ Объем адресуемой оперативной памяти (RAM) равен 256 байтам.
  - ❑ Адресуется память путем последовательного занесения в нее байтов.

# Ввод кода программы

Введите команды программы пересылки кодов в регистры общего назначения

Для увеличения - уменьшения размеров шрифта используют кнопки



**Общий синтаксис команды:  
MOV A1,A2**

**Схема работы команды:**

**R <- Code - Запись кода в регистр  
R->[Addr] – Запись кода из регистра в  
оперативную память по адресу Addr  
R<-[Addr] – Запись кода из ячейки  
оперативной памяти с адресом Addr в  
регистр.**

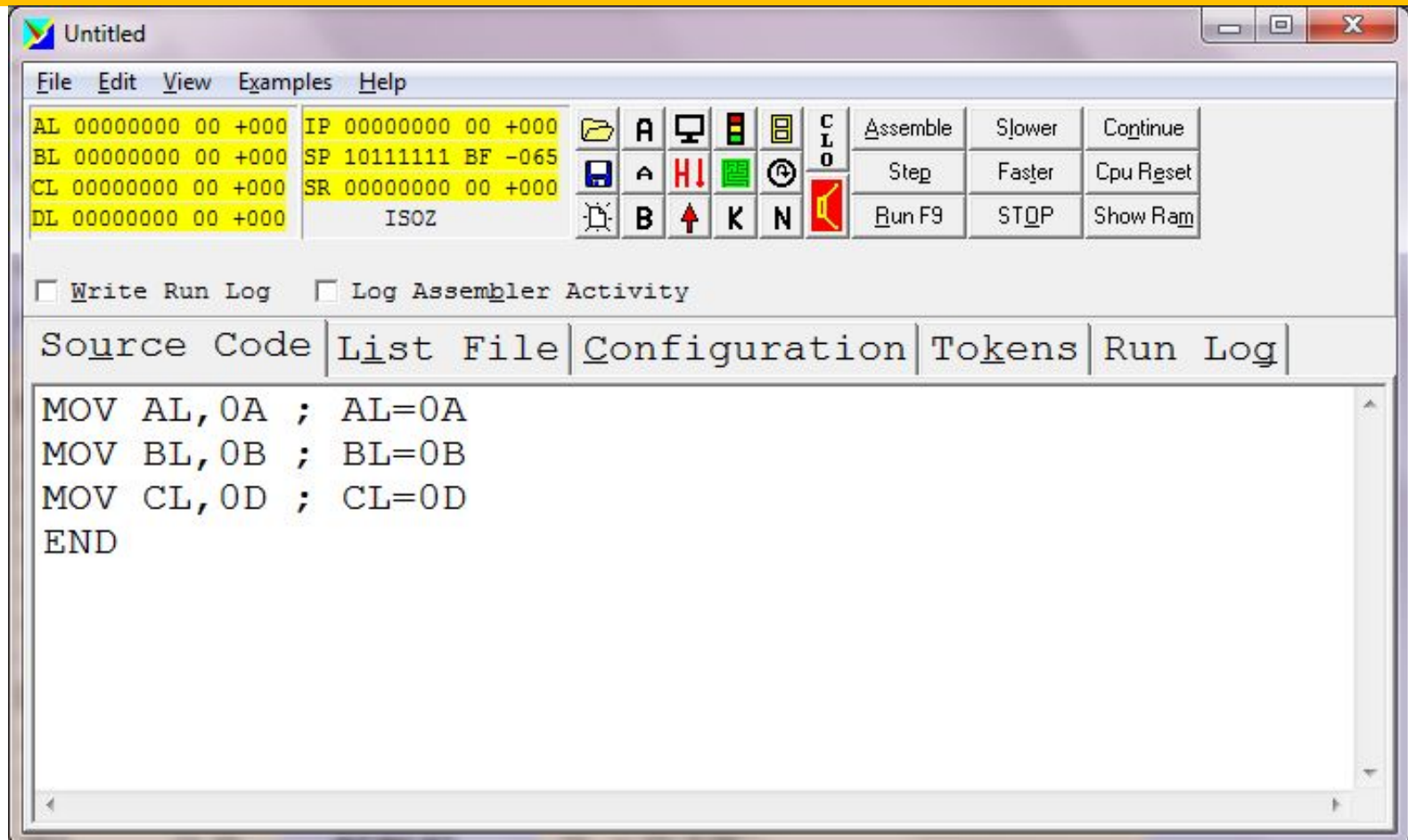
# Примеры использования команды

---

- ❑ **MOV AL,12; Запись в регистр кода 12**
- ❑ **MOV [17],CL ;Запись в ячейку памяти с адресом 17 ;содержания регистра CL**
- ❑ **MOV BL,[C3] ; Запись в регистр значения из ячейки ;с адресом C3**
- ❑ **MOV [CL],DL ;Запись содержания регистра ;ячейку адрес которой хранится в регистре CL**
- ❑ **MOV AL,[BL] ;Запись содержания ячейки по ;адресу, который хранится в регистре BL в регистр AL**

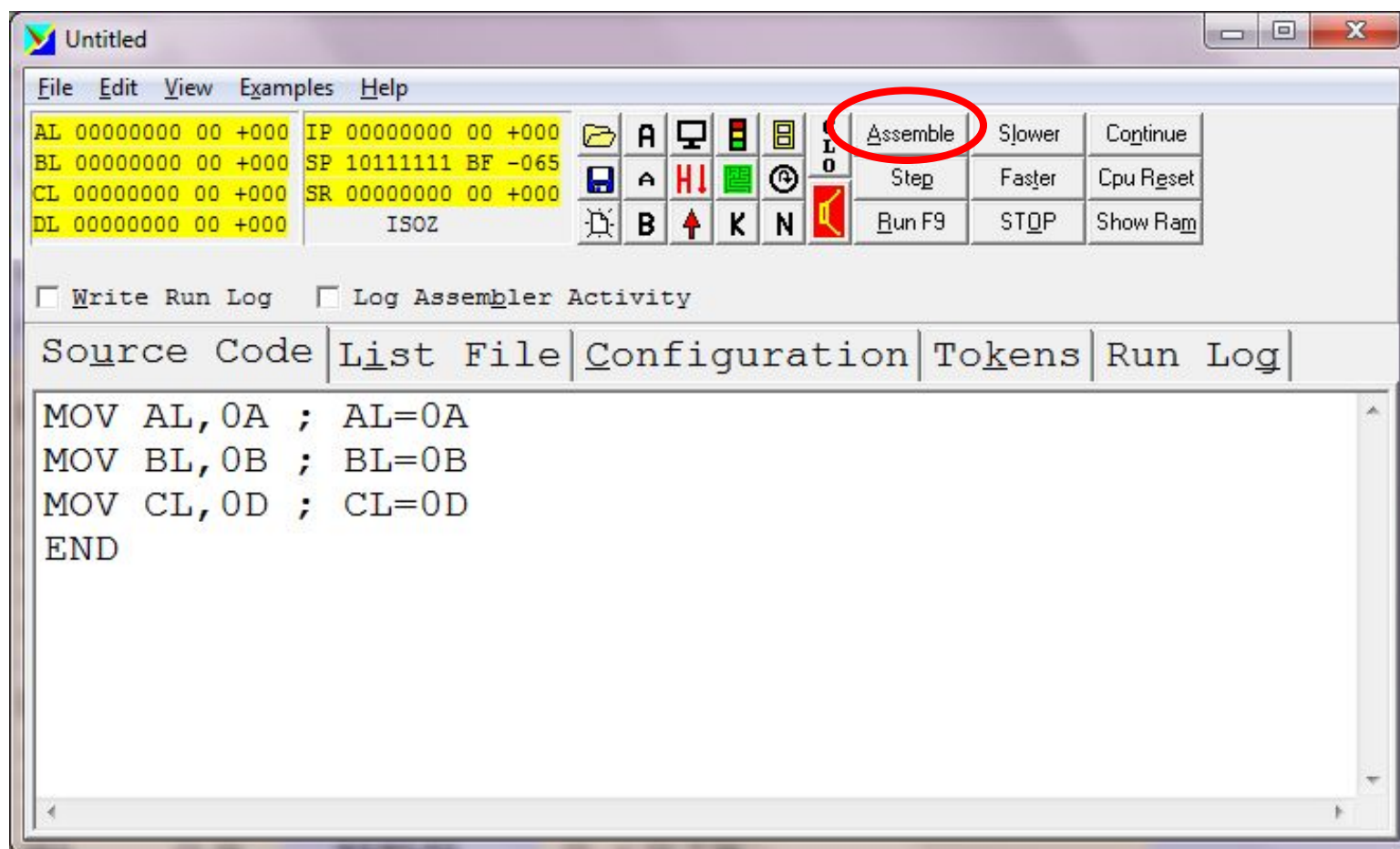
# Редактор кода

- ❑ Команда END не является ассемблерной командой это команда эмулятора, означающая конец программы.
- ❑ Для ввода комментариев используется специальный символ «;».
- ❑ Коды при записи в регистр задаются в шестнадцатеричном формате!
- ❑ Регистр символов кода и команд не имеет значения



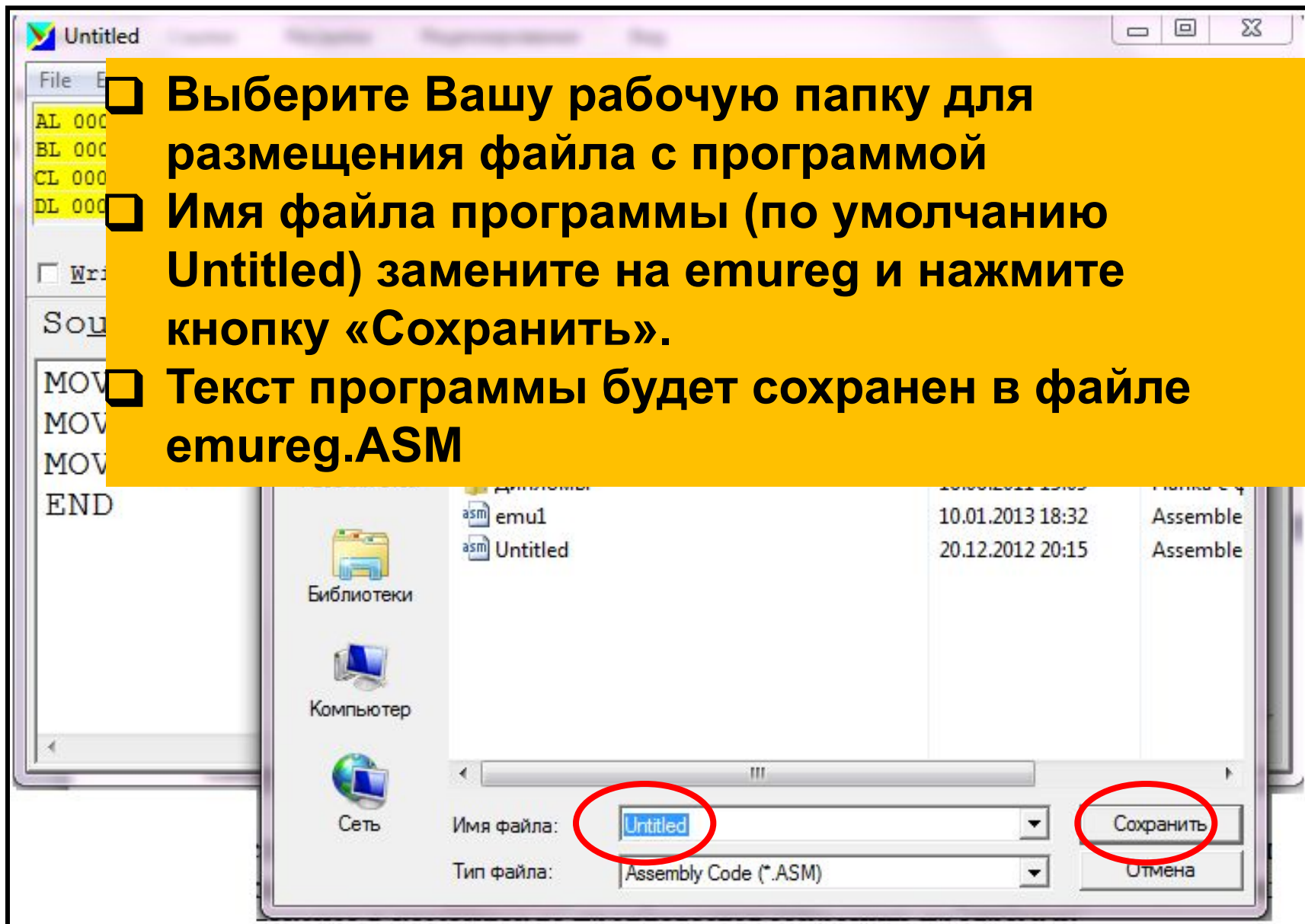


После набора команд программы выполняется ассемблирование программы путем нажатия кнопки «Assemble» в верхней части окна

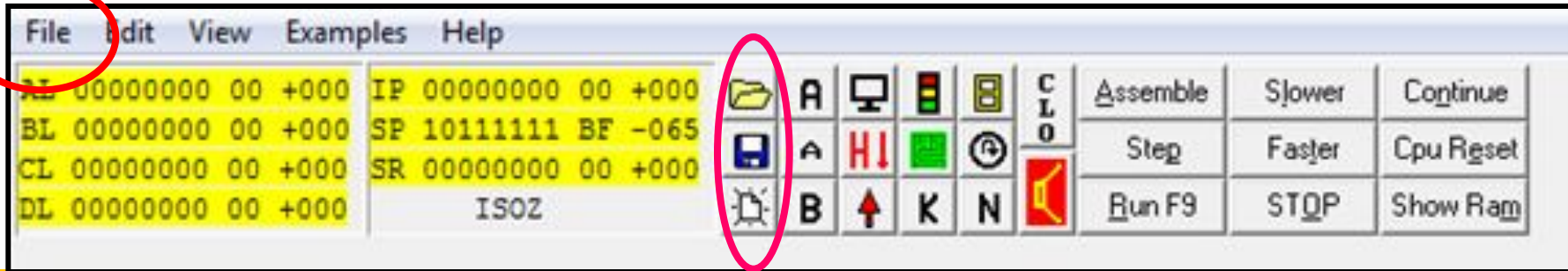


# Сохранение программы

- ❑ Выберите Вашу рабочую папку для размещения файла с программой
- ❑ Имя файла программы (по умолчанию Untitled) замените на emureg и нажмите кнопку «Сохранить».
- ❑ Текст программы будет сохранен в файле emureg.ASM



# Команды для работы с текстом программы



Для работы с файлами можно использовать команды пункта меню File:

**Open** – Открыть файл.

**Save** – Сохранить файл.

**Save As** – Сохранить файл под новым именем.

	Открыть файл
	Сохранить файл
	Создать файл

# Адресное пространство программы

После сохранения программа будет скомпилирована в оперативную память и будет готова к исполнению.

The screenshot shows an assembler interface with the following components:

- Registers and Values:**

AL	00000000	00	+000	IP	00000000	00	+000
BL	00000000	00	+000	SP	10111111	BF	-065
CL	00000000	00	+000	SR	00000000	00	+000
DL	00000000	00	+000		ISOZ		
- Buttons:** Assemble, Slower, Continue, Step, Faster, Cpu Riset, Run F9, STQP, Show Ram.
- Source Code:**

```
MOV AL, 0A ; AI=0A
MOV BL, 0B ;
MOV CL, 0D ;
END
```
- RAM Source Code View:**

	0	1	2	3	4	5	7	8	9	A	B	C	D	E	F
00	MOV	AL	0A	MOV	BL	0B	MOV	CL	0D	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0															
D0															
E0															
F0															
- Footer:** Hexadecimal, ASCII, Source (selected).

# Программа в памяти

**Наша программа занимает диапазон адресов 00 до 09. Не забывайте, что значения шестнадцатеричные !**

Source Code | List File | Configuration | Tokens | Run Log

```

MOV AL, 0A ; AT=0A
MOV BL, 0B ;
MOV CL, 0D ;
END

```

RAM Source Code View

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	MOV	AL	0A	MOV	BL	0B	MOV	CL	0D	END	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0																
D0																
E0																
F0																END

Hexadecimal
  ASCII
  Source

# Исполнение программы

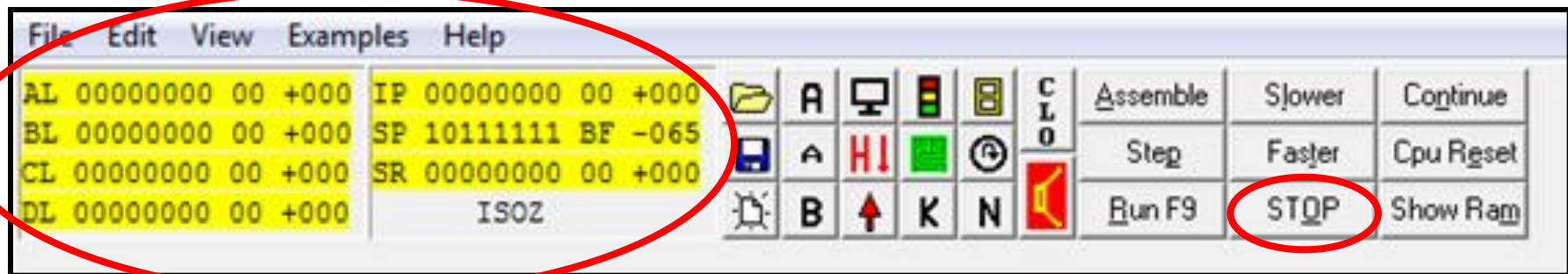
Для выполнения программы используем режим пошагового выполнения. Нажимайте последовательно кнопку «Step» и следите как изменяются значения в регистрах процессора.

The screenshot shows a debugger window with a menu bar (File, Edit, View, Examples, Help) and a control panel. The register window on the left is highlighted with a red box and contains the following data:

AL	00000000	00	+000	IP	00000000	00	+000
BL	00000000	00	+000	SP	10111111	BF	-065
CL	00000000	00	+000	SR	00000000	00	+000
DL	00000000	00	+000				ISOZ

The control panel on the right contains several buttons: Assemble, Slower, Continue, Step (circled in red), Faster, Cpu Reset, Run F9, STQP, and Show Ram.

# Выполнение программы

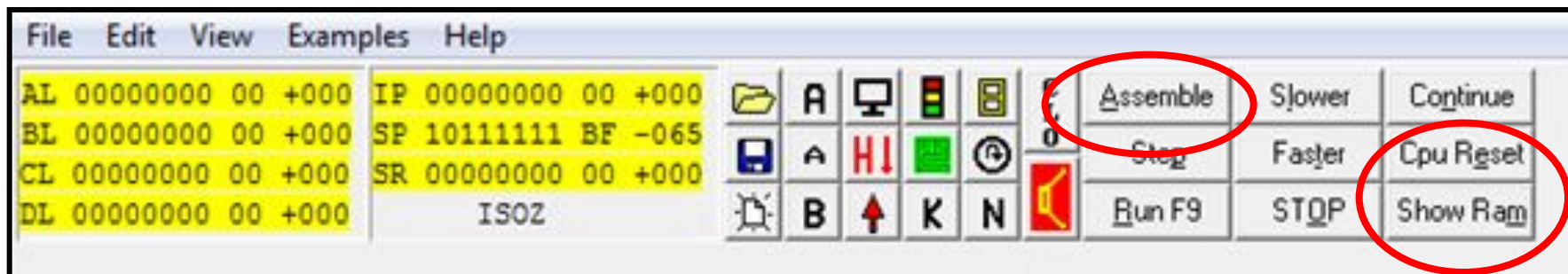


Обратите внимание, что значение регистров выводится в трех системах исчисления двоичной, шестнадцатеричной и десятичной.

После достижения команды END нажмите кнопку «STOP».

# Перезапуск программы

В любом режиме исполнения программы для повторного ее исполнения следует нажать кнопку «CPU Reset!».

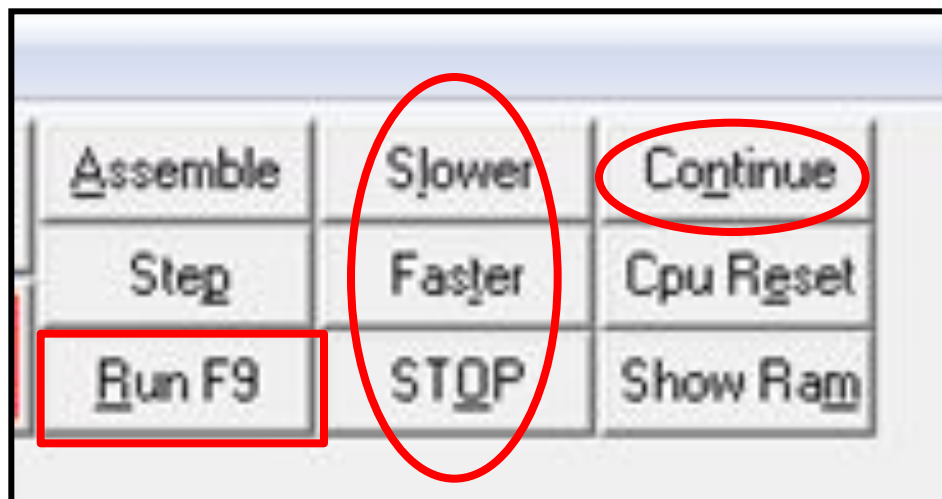


- ❑ Если окно отображения оперативной памяти закрыто, его можно вывести, нажав кнопку «Show Ram».
- ❑ Сброс оперативной памяти выполняется после нажатия кнопки «Assemble».

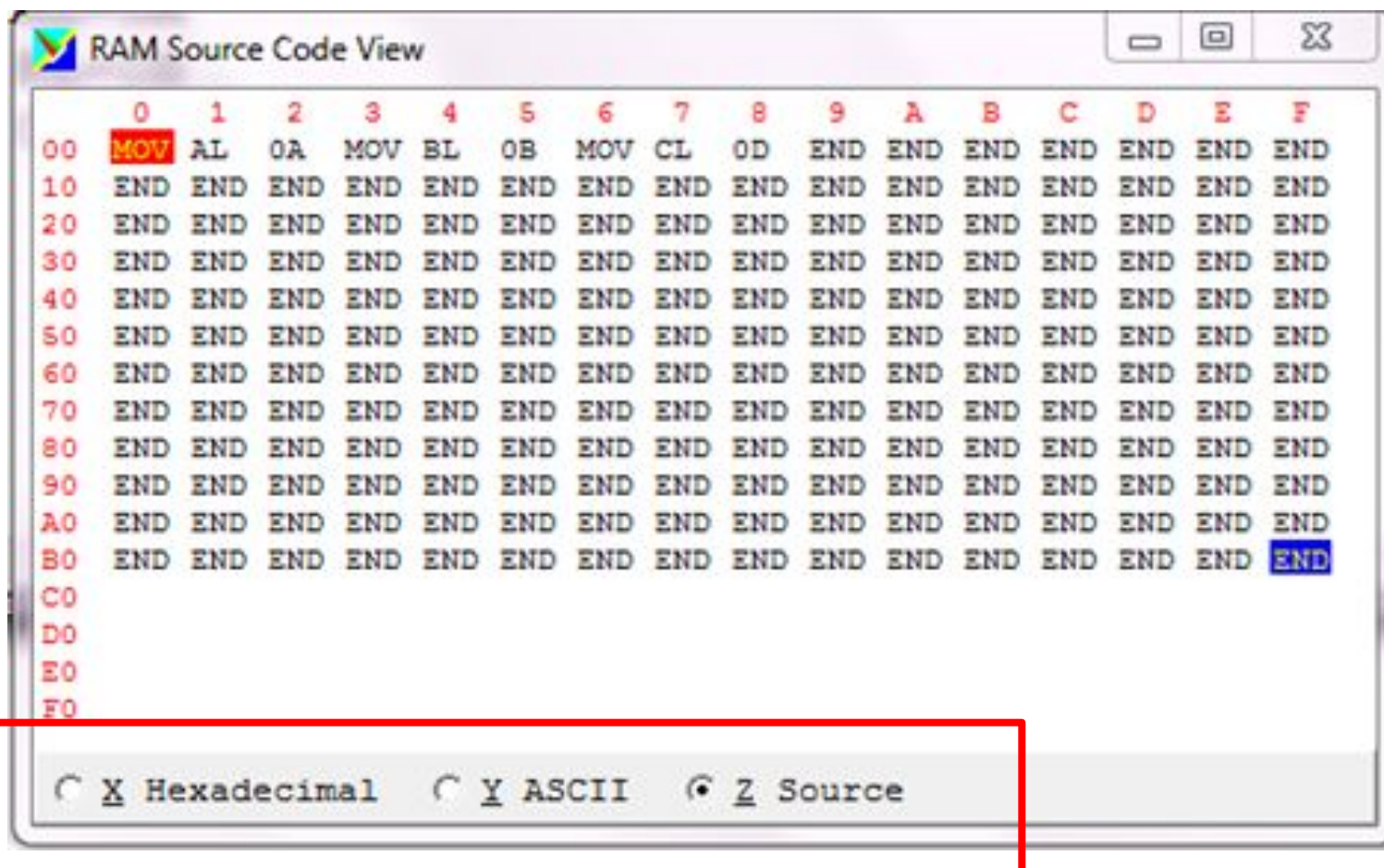


# Скорость исполнения программы

- ❑ Программу можно выполнить в автоматическом режиме используя кнопку «Run». В автоматическом режиме исполнения можно использовать кнопки:
- ❑ STOP – Приостановить выполнение программы.
- ❑ Continue – Продолжить выполнение.
- ❑ Slower – уменьшить частоту процессора эмулятора, для замедления обработки команд.
- ❑ Faster – увеличить частоту работы процессора.



Содержание оперативной памяти можно выводить в одном из трех режимов

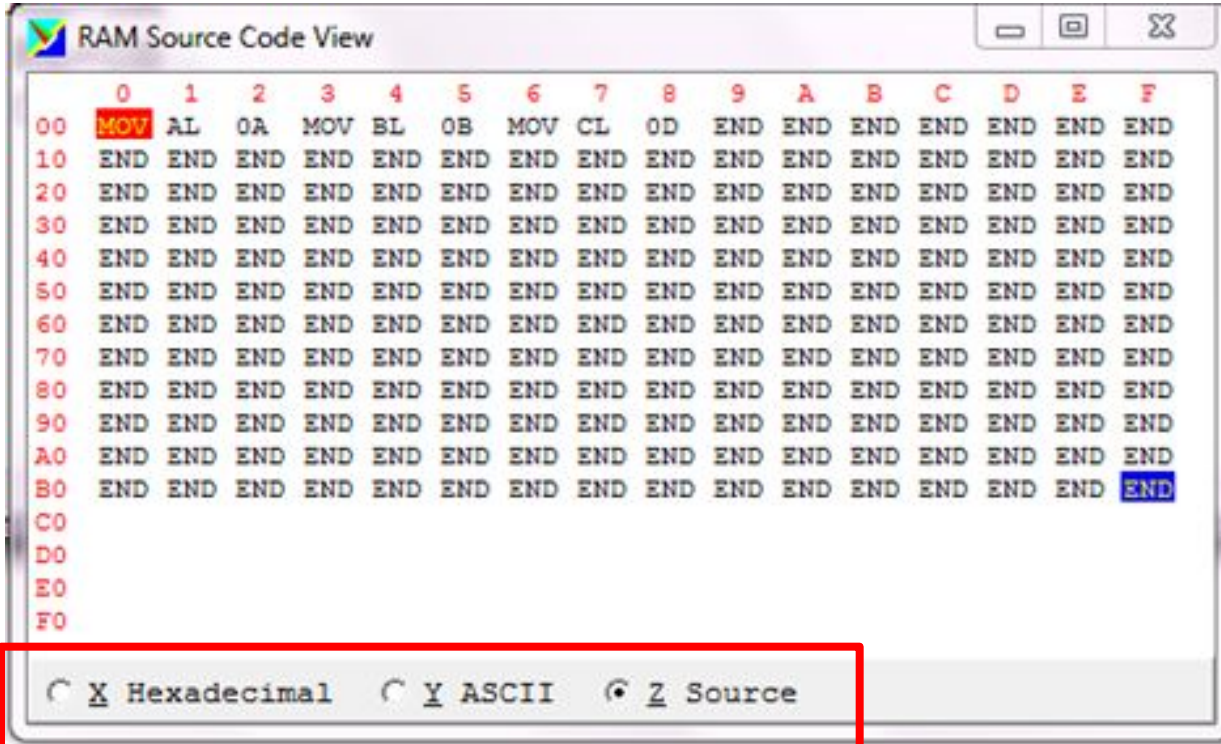


The screenshot shows a window titled "RAM Source Code View" with a menu bar containing minimize, maximize, and close buttons. The main area displays a memory dump with columns for hexadecimal addresses (00 to F0) and source code instructions. The instruction at address 00 is "MOV AL, 0A", and the instruction at address B0 is "END". The "END" instruction at B0 is highlighted with a blue selection box. At the bottom of the window, there is a mode selection bar with three radio buttons: "Hexadecimal", "ASCII", and "Source". The "Source" mode is selected, and this bar is enclosed in a red rectangular box.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	MOV	AL	0A	MOV	BL	0B	MOV	CL	0D	END	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0																
D0																
E0																
F0																

Hexadecimal    ASCII    Source

# Ячейки оперативной памяти



The screenshot shows a window titled "RAM Source Code View" with a grid of memory addresses and assembly instructions. The columns are labeled with hexadecimal digits 0 through F. The rows are labeled with hexadecimal addresses from 00 to F0. The first row (00) contains the instruction "MOV AL 0A MOV BL 0B MOV CL 0D END END END END END END". The last row (B0) contains "END" and is highlighted with a blue selection box. At the bottom, a control bar is highlighted with a red box, containing three radio buttons: "X Hexadecimal", "Y ASCII", and "Z Source". The "Z Source" option is selected.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	MOV	AL	0A	MOV	BL	0B	MOV	CL	0D	END	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0																
D0																
E0																
F0																

X Hexadecimal    Y ASCII    Z Source

- Hexadecimal – побайтовый просмотр в шестнадцатеричном формате.
- ASCII – просмотр содержания оперативной памяти в символьном виде.
- Source – вывод программы в оперативной памяти (используется по умолчанию).

**Упражнение № 1. Программа change1.asm.  
Напишите программу для обмена значений регистров AL и CL, используя оперативную память. Исходное значение кодов в регистрах AL=1F, CL=2F.**

**Псевдо код программы:**

**AL <= 1F**

**CL <= 2F**

**Адрес 52 <= AL**

**Адрес 53 <= CL**

**Адрес 52 => CL**

**Адрес 53 => AL**

**КОНЕЦ**

- ❑ В эмуляторе участок оперативной памяти в диапазоне адресов от C0 до FF представляет собой видеопамять.
- ❑ Если в ячейку занести ASCII код символа, то автоматически открывается устройство VDU (Visual Display Unit) которое отображает содержание видеопамети в виде СИМВОЛОВ.

# Кодировка символов

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

RAM Source Code View

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	MOV	AL	0A	MOV	BL	0B	MOV	CL	0D	END	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0																
D0																
E0																
F0																

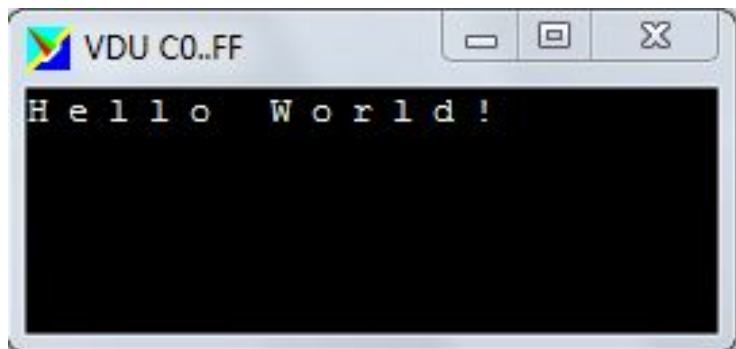
Видеопамять

Hexadecimal  ASCII  Source

# Вывод сообщения

---

**Упражнение № 2. Программа hello.asm** Напишите программу для вывода на устройство VDU контрольного сообщения Hello World!.



***Следует иметь в виду, что оперативная память обновляется только после повторного ассемблирования программы, кнопка «Assemble» !***



# Запись кодов в видео память

AL <= 48  
Адрес C0 <= AL  
AL <= 65  
Адрес C1 <= AL  
AL <= 6C  
Адрес C2 <= AL  
AL <= 6C  
Адрес C3 <= AL  
AL <= 6F  
Адрес C4 <= AL  
AL <= 20  
Адрес C5 <= AL  
AL <= 57  
Адрес C6 <= AL

Псевдокод  
программы

AL <= 6F  
Адрес C7 <= AL  
AL <= 72  
Адрес C8 <= AL  
AL <= 6C  
Адрес C9 <= AL  
AL <= 64  
Адрес CA <= AL  
AL <= 21  
Адрес CB <= AL  
Конец

# Программный стек

---

- ❑ **Стек – участок оперативной памяти доступ, к которому построен по принципу:  
*«Первый пришел, последний вышел» !***
- ❑ **Используется стек для временного хранения кодов в процессе работы программы.**

**Команды для работы со стеком:**

**PUSH R ;Запись содержания регистра в стек**

**POP R ;Извлечение кода с верхушки стека в регистр**

- ❑ **Стек располагается в нижнем диапазоне адресов оперативной памяти и начинается с адреса BF.**
- ❑ **Адресуется стек с помощью регистра SP.**

**Упражнение № 3. Напишите программу change2.asm в соответствии с упражнением 1, используя стек для временного хранения данных.**

**Псевдо код программы:**

**AL <= 1F**

**CL <= 2F**

**AL ↓**

**CL ↓**

**AL ↑**

**CL ↑**

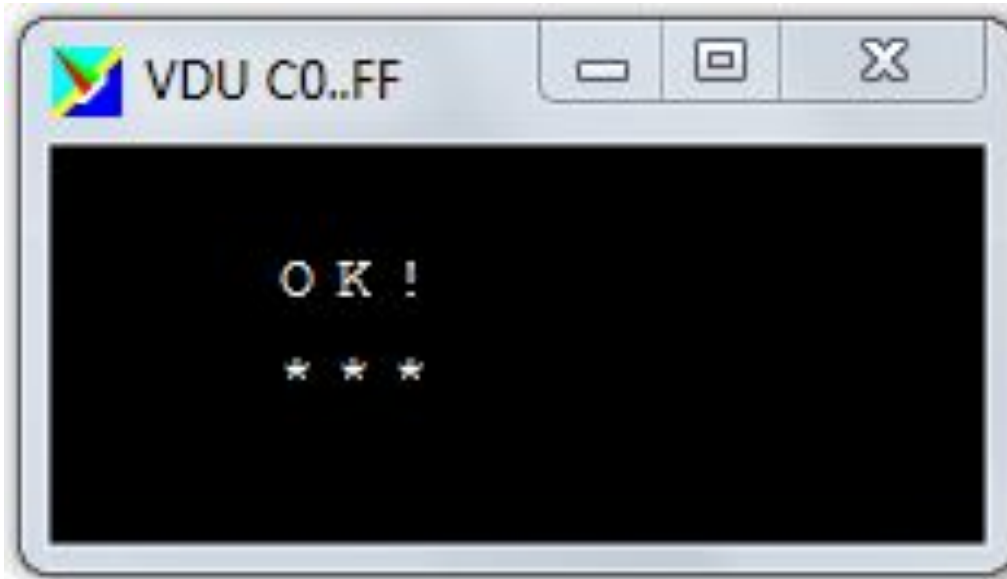
**КОНЕЦ**

**Примечание:**

↓ - поместить в стек

↑ - извлечь из стека

**Контрольное задание №1. Программа messages.asm** Напишите программу для вывода сообщения.



**Контрольное задание №2. Напишите программу revers.asm.**

- Занесите в стек коды символов строки “QWERTY”, извлеките коды из стека и выведите на устройство VDU.**
- Объясните результат.**