Системне програмування, частина 1

ЛАБОРАТОРНА РОБОТА № 1.

Ознайомлення з середовищем MASM32. Структура програми на асемблері. Програмування вводу та виводу.

Мета: освоїти послідовність дій для компіляції програм, написаних на мові Assembler за допомогою ссередовищ MASM32 і Visual Studio 2019. Навчитися реалізовувати ввід з клавіатури та вивід на екран символьних даних.

Програма мовою асемблера має таку структуру:

```
.686
.model flat, stdcall
option casemap: none
.data <ініціалізовані дані>
.data? <неініціалізовані дані>
.const <константи>
.code <мітка> <код>
end <мітка>
```

Директива .686 вказує компілятору асемблера, що необхідно використовувати набір операцій процесора певного покоління.

Директива .model дозволяє вказувати модель пам'яті та угоду про виклики. Архітектура Win32 використовує лише одну модель пам'яті – flat, що й зазначено у наведеному прикладі. Угоди про виклики визначають порядок передачі параметрів та порядок очищення стека.

Директива option casemap: none змушує компілятор мови асемблера розрізняти великі та маленькі літери у мітках та іменах процедур.

Директиви .data, .data?, .const та .code визначають те, що називається секціями. У Win32 немає сегментів, але адресний простір можна розділити на логічні секції. Початок однієї секції відзначає кінець попередньої. Є дві групи секцій: даних та коду.

Секція .data містить ініціалізовані дані програми.

Секція .data? містить неініціалізовані дані програми. Іноді потрібно лише попередньо виділити пам'яь, не ініціалізуючи її. Ця секція при цьому і призначається. Перевага неініціалізованих даних у тому, що вони не займають місця у файлі. Ви лише повідомляєте компілятору, скільки місця вам знадобиться, коли програма завантажиться в пам'ять.

Секція .const містить оголошення констант, що використовуються програмою. Константи не можуть бути змінені. Спроба змінити константу викликає аварійне завершення програми.

Задіяти усі три секції не обов'язково.

Є лише одна секція для коду: .code. У ньому міститься весь код.

<мітка> та end <мітка> встановлюють межі коду. Обидві мітки мають бути ідентичними. Весь код повинен розташовуватись між цими рядками. Будь-яка програма під Windows має, як мінімум, коректно завершитись. Для цього потрібно викликати функцію Win32 API ExitProcess.

Для написання, компілювання і виконання програм, написаних на мові Assembler необхідно встановити середовище MASM32 (рис. 1.1). Середовище має простий текстовий редактор, за допомогою якого можна набирати тексти програм. У меню Project є можливість відкомпілювати програму, зробити .exe файл і виконати його.

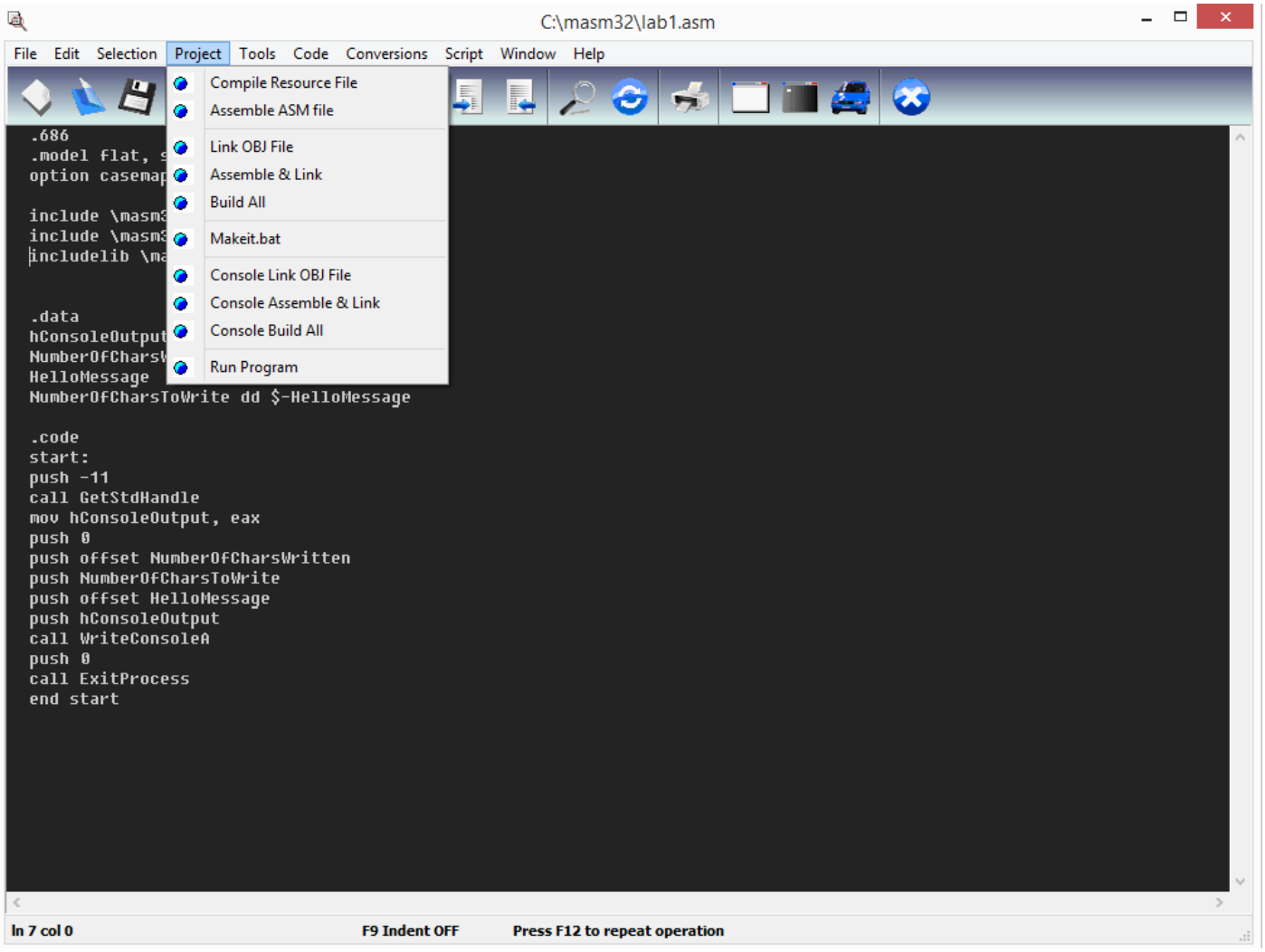

Рис. 1.1. Середовище MASM32.

Також для написання, налагодження і запуску програм, написаних на мові Assembler можна скористатись середовищем Microsoft Visual Studio 2019. Створюємо порожній проект і у властивостях проекту в розділі «Залежності збірки» ставимо галочку «masm» (рис. 1.2 і 1.3) .

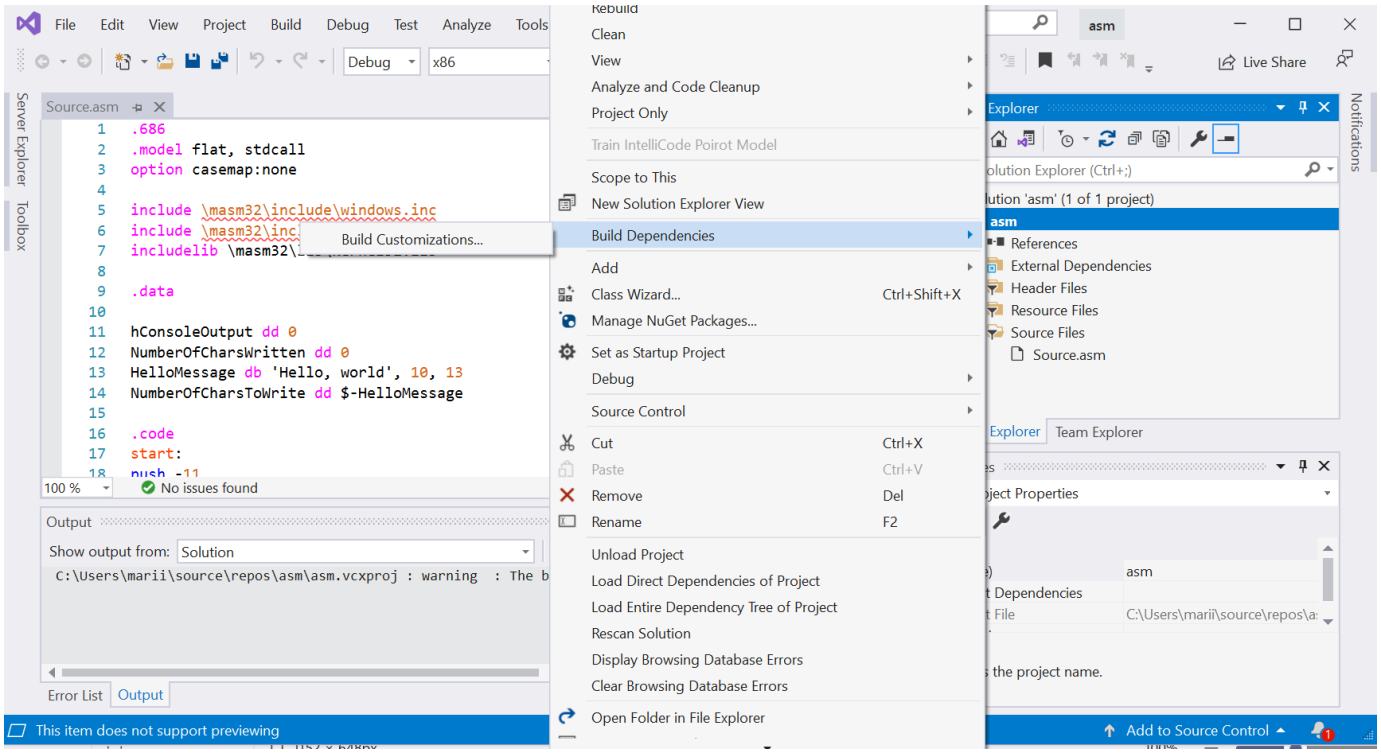

Рис. 1.2. Налаштування середовища Visual Studio 2019 – «залежності збірки».

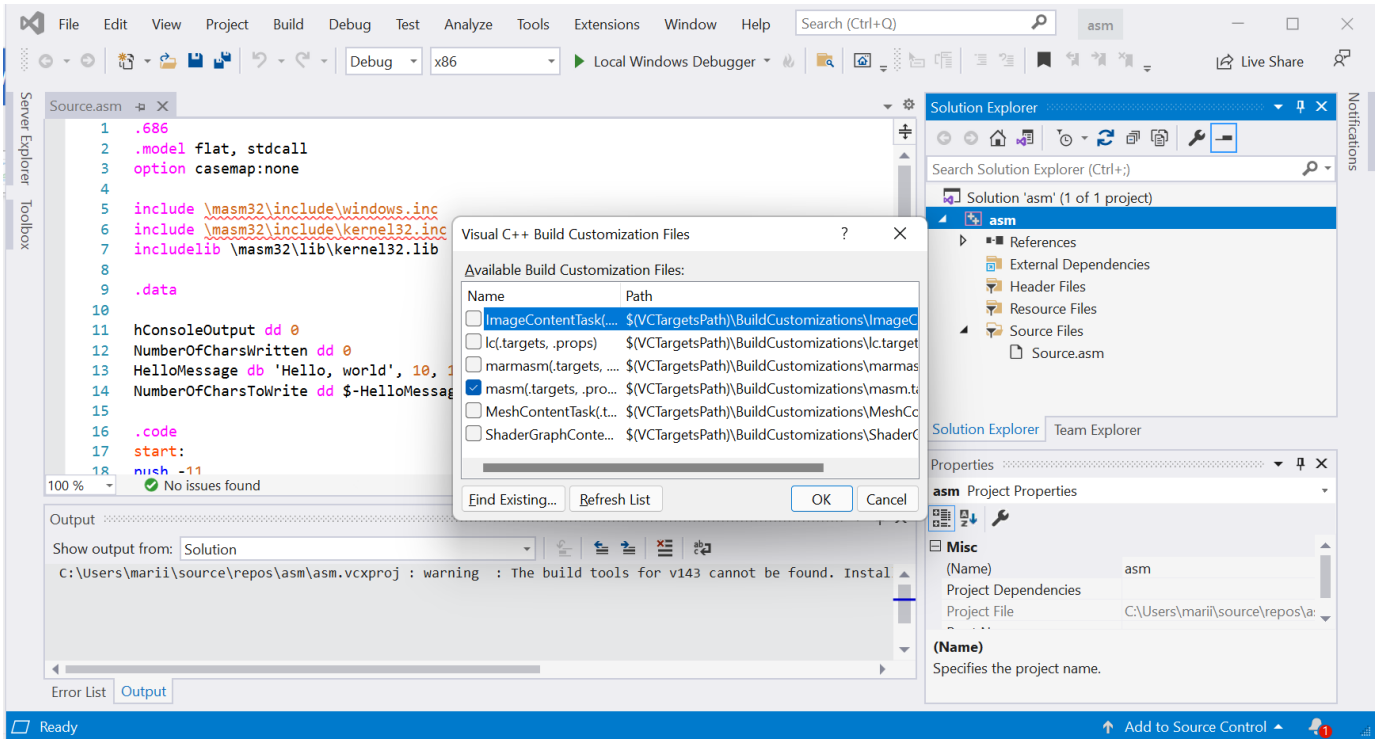

Рис. 1.3. Налаштування середовища Visual Studio 2019 – «галочка masm».

Після цього створюємо і додаємо до проекту файл з розширенням .asm. У властивостях цього файлу має бути тип «Microsoft Macro Assembler» (рис. 1.4).

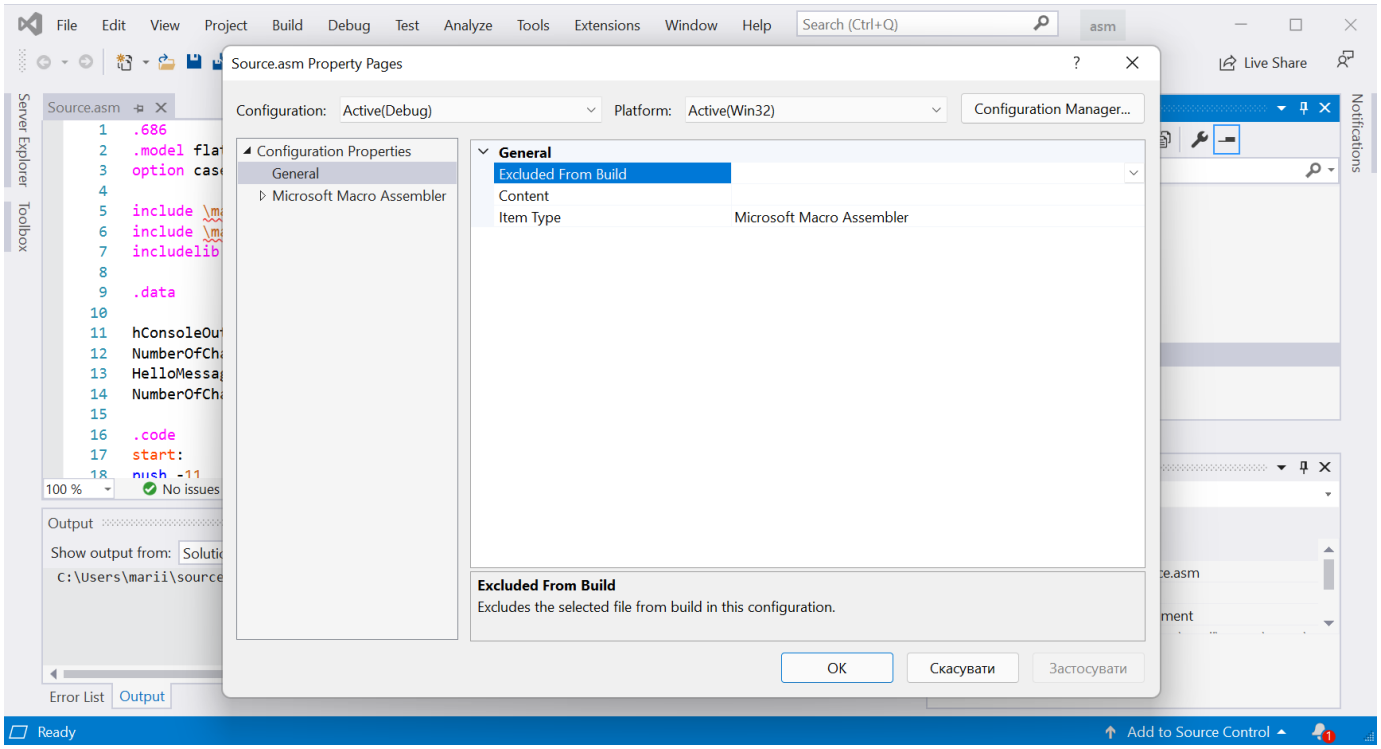

Рис. 1.4. Налаштування середовища Visual Studio 2019 – «властивості файлу .asm».

Під час виконання програми у середовищі Visual Studio 2019 в покроковому режимі є можливість налагоджувати програму відслідковуючи стан регістрів процесора. Для цього в покроковому режимі виконання програми необхідно в меню Debug/Windows вибрати Registers (рис. 1.5).

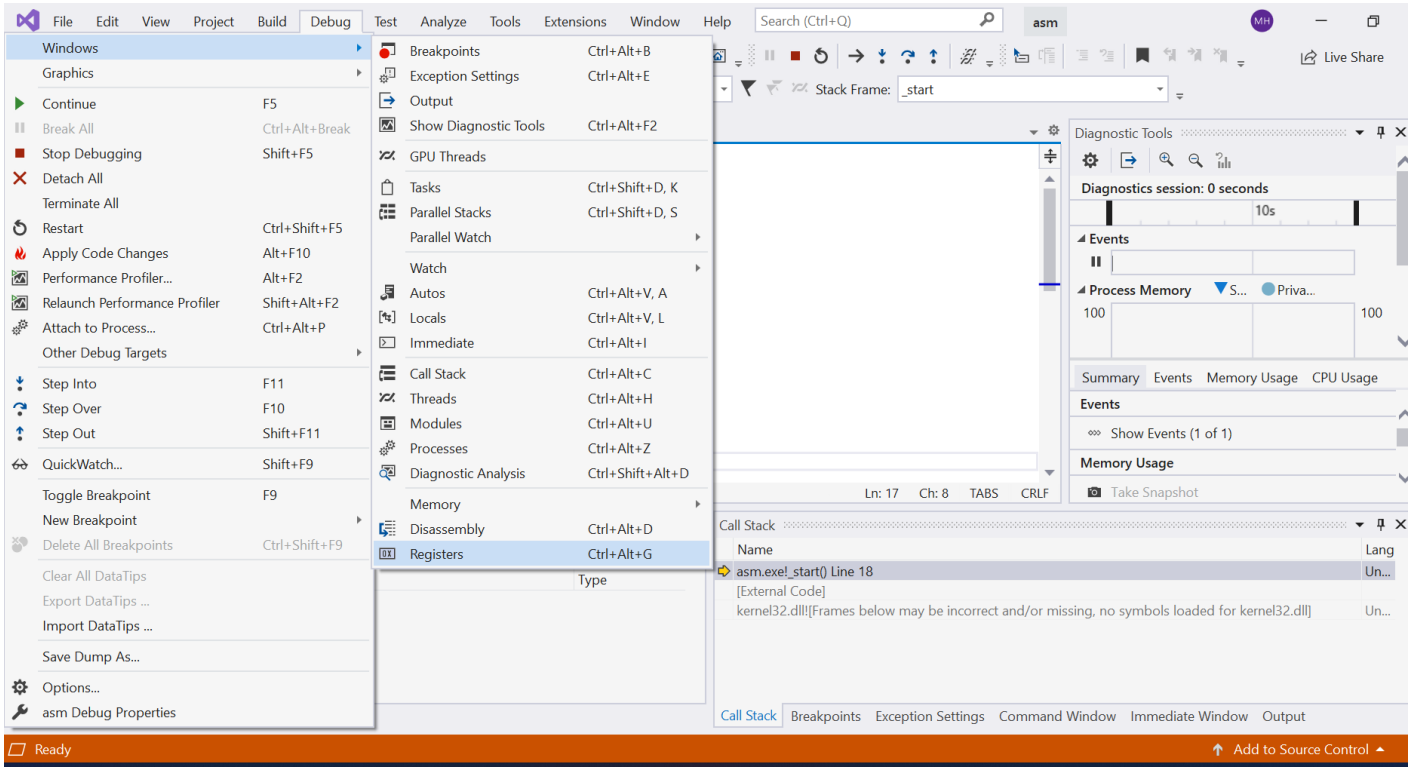

Рис. 1.5. Вибір вікна для відображення стану регістрів.

Правою кнопкою миші можна вибрати певну групу регістрів для відображення (рис. 1.6).

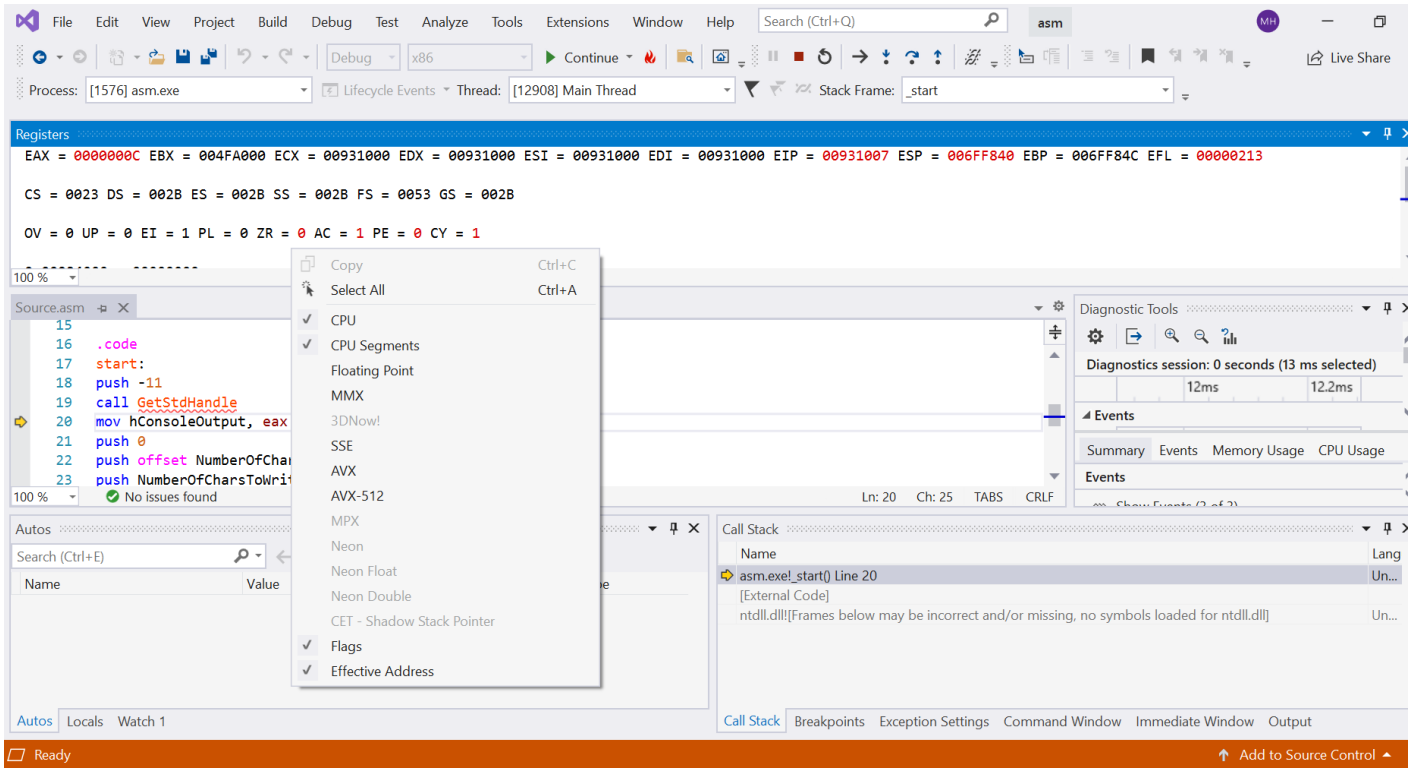

Рис. 1.5. Вікно відображення стану регістрів.

ЗАВДАННЯ:

1. За допомогою текстового редактора створити файл **<прізвище>.asm** , який містить програму, приведену нижче. (Параметр <прізвище> повинен містити Ваше прізвище в англійській транслітерації і мати довжину до 8 символів).
2. В тексті програми змінити повідомлення, що міститься за міткою HelloMessage так, щоб воно відображало Ваше прізвище. Зберегти внесені зміни.
3. Створити **<прізвище>.exe**-файл засобами системи MASM32 або Visual Studio 2019.
4. Виконати створену програму і переконатися, що вона працює коректно, тобто виводить Ваше прізвище на екран.

```
.686
.model flat, stdcall
option casemap:none

include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib

.data
hConsoleOutput dd 0
NumberOfCharsWritten dd 0
HelloMessage db 'Hello, world', 10, 13
NumberOfCharsToWrite dd $-HelloMessage

.code
start:
push -11
call GetStdHandle
mov hConsoleOutput, eax
push 0
push offset NumberOfCharsWritten
push NumberOfCharsToWrite
push offset HelloMessage
push hConsoleOutput
call WriteConsoleA
push 0
call ExitProcess
end start
```

5. Створити програму, яка в текстовому режимі виводить **за допомогою псевдографіки** зображення заданого варіантом символу.
6. Перевірити роботу створеної програми і продемонструвати її виконання.
7. Скласти звіт про виконану роботу (з приведенням тексту програми).

ВАРІАНТИ ЗАВДАННЯ:

варіант	символ	варіант	символ
1	**A**	16	**λ**
2	**F**	17	**X**
3	**H**	18	**Z**
4	**N**	19	**Д**
5	**M**	20	**Ш**
6	**K**	21	**Е**
7	**L**	22	**У**
8	**V**	23	**Ц**
9	**Y**	24	**И**
10	**α**	25	**ω**
11	**β**	26	**τ**
12	**δ**	27	**ρ**
13	**Я**	28	**Ж**
14	**Ї**	29	**G**
15	**Ю**	30	**W**

Повна таблиця ASCII символів (кодова таблиця CP866)

С–символ;10–десяткове значення;16–шістнадцяткове значення

С	*10*	*16*	*С*	*10*	*16*	*С*	*10*	*16*	*С*	*10*	*16*	*С*	*10*	*16*	*С*	*10*	*16*	*С*	*10*	*16*	*С*	*10*	*16*
	0	00	**пробіл**	32	20	**@**	64	40	**`**	96	60	**А**	128	80	**а**	160	A0	└	192	C0	**р**	224	E0
	1	01	**!**	33	21	**A**	65	41	**a**	97	61	**Б**	129	81	**б**	161	A1	┴	193	C1	**с**	225	E1
	2	02	**"**	34	22	**B**	66	42	**b**	98	62	**В**	130	82	**в**	162	A2	┬	194	C2	**т**	226	E2
	3	03	**#**	35	23	**C**	67	43	**c**	99	63	**Г**	131	83	**г**	163	A3	├	195	C3	**у**	227	E3
	4	04	**$**	36	24	**D**	68	44	**d**	100	64	**Д**	132	84	**д**	164	A4	─	196	C4	**ф**	228	E4
	5	05	**%**	37	25	**E**	69	45	**e**	101	65	**Е**	133	85	**е**	165	A5	┼	197	C5	**х**	229	E5
	6	06	**&**	38	26	**F**	70	46	**f**	102	66	**Ж**	134	86	**ж**	166	A6	╞	198	C6	**ц**	230	E6
	7	07	**'**	39	27	**G**	71	47	**g**	103	67	**З**	135	87	**з**	167	A7	╟	199	C7	**ч**	231	E7
	8	08	**(**	40	28	**H**	72	48	**h**	104	68	**И**	136	88	**и**	168	A8	╚	200	C8	**ш**	232	E8
	9	09	**)**	41	29	**I**	73	49	**i**	105	69	**Й**	137	89	**й**	169	A9	╔	201	C9	**щ**	233	E9
	10	0A	*****	42	2A	**J**	74	4A	**j**	106	6A	**К**	138	8A	**к**	170	AA	╩	202	CA	**ъ**	234	EA
	11	0B	+	43	2B	**K**	75	4B	**k**	107	6B	**Л**	139	8B	**л**	171	AB	╦	203	CB	**ы**	235	EB
	12	0C	**,**	44	2C	**L**	76	4C	**l**	108	6C	**М**	140	8C	**м**	172	AC	╠	204	CC	**ь**	236	EC
	13	0D	-	45	2D	**M**	77	4D	**m**	109	6D	**Н**	141	8D	**н**	173	AD	═	205	CD	**э**	237	ED
	14	0E	**.**	46	2E	**N**	78	4E	**n**	110	6E	**О**	142	8E	**о**	174	AE	╬	206	CE	**ю**	238	EE
	15	0F	/	47	2F	**O**	79	4F	**o**	111	6F	**П**	143	8F	**п**	175	AF	╧	207	CF	**я**	239	EF
	16	10	**0**	48	30	**P**	80	50	**p**	112	70	**Р**	144	90	░	176	B0	╨	208	D0	**Ё**	240	F0
	17	11	**1**	49	31	**Q**	81	51	**q**	113	71	**С**	145	91	▒	177	B1	╤	209	D1	**ё**	241	F1
	18	12	**2**	50	32	**R**	82	52	**r**	114	72	**Т**	146	92	▓	178	B2	╥	210	D2	**Ґ**	242	F2
	19	13	**3**	51	33	**S**	83	53	**s**	115	73	**У**	147	93	│	179	B3	╙	211	D3	**ґ**	243	F3
	20	14	**4**	52	34	**T**	84	54	**t**	116	74	**Ф**	148	94	┤	180	B4	╘	212	D4	**Є**	244	F4
	21	15	**5**	53	35	**U**	85	55	**u**	117	75	**Х**	149	95	╡	181	B5	╒	213	D5	**є**	245	F5
	22	16	**6**	54	36	**V**	86	56	**v**	118	76	**Ц**	150	96	╢	182	B6	╓	214	D6	**І**	246	F6
	23	17	**7**	55	37	**W**	87	57	**w**	119	77	**Ч**	151	97	╖	183	B7	╫	215	D7	**і**	247	F7
	24	18	**8**	56	38	**X**	88	58	**x**	120	78	**Ш**	152	98	╕	184	B8	╪	216	D8	**Ї**	248	F8
	25	19	**9**	57	39	**Y**	89	59	**y**	121	79	**Щ**	153	99	╣	185	B9	┘	217	D9	**ї**	249	F9
	26	1A	**:**	58	3A	**Z**	90	5A	**z**	122	7A	**Ъ**	154	9A	║	186	BA	┌	218	DA	**·**	250	FA
	27	1B	**;**	59	3B	**[**	91	5B	**{**	123	7B	**Ы**	155	9B	╗	187	BB	█	219	DB	√	251	FB
	28	1C	<	60	3C	\	92	5C	|	124	7C	**Ь**	156	9C	╝	188	BC	▄	220	DC	**№**	252	FC
	29	1D	=	61	3D	**]**	93	5D	**}**	125	7D	**Э**	157	9D	╜	189	BD	▌	221	DD	¤	253	FD
	30	1E	>	62	3E	^	94	5E	~	126	7E	**Ю**	158	9E	╛	190	BE	▐	222	DE	■	254	FE
	31	1F	**?**	63	3F	**_**	95	5F		127	7F	**Я**	159	9F	┐	191	BF	▀	223	DF		255	FF

Приклад виконання.

Завдання: за допомогою псевдографіки вивести зображення символу «О».

Для вирішення поставленої задачі, слід визначити коди ASCII псевдографічних символів. Для цього необхідно скористатися повною таблицею ASCII символів, приведеною вище.

З таблиці видно, що псевдографічні символи розташовані між 176 і 223 кодом. Оберемо для створення границь зображення заданого символу заповнений прямокутник (код 219), а для наповнення – розріджений прямокутник (код 176).

```
.686
.model flat, stdcall
option casemap:none

include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib

.data
hConsoleOutput dd 0
NumberOfCharsWritten dd 0
Symbol   db  60 dup (32),10,13
   db  19 dup (32),26 dup(219),19 dup (32), 10, 13
   db  17 dup (32),2 dup(219),7 dup(176),2 dup(219),8 dup(32),2 dup(219),7 dup(176),2
dup(219),17 dup(32), 10, 13
   db  15 dup (32),2 dup(219),7 dup(176),2 dup(219),12 dup(32),2 dup(219),7 dup(176),2
dup(219),15 dup(32), 10,13
   db  13 dup (32),2 dup(219),7 dup(176),2 dup(219),16 dup(32),2 dup(219),7 dup(176),2
dup(219),13 dup(32), 10,13
   db  11 dup (32),2 dup(219),7 dup(176),2 dup(219),20 dup(32),2 dup(219),7 dup(176),2
dup(219),11 dup(32), 10,13
   db  8 dup( 9 dup (32),2 dup(219),7 dup(176),2 dup(219),24 dup(32),2 dup(219),7
dup(176),2 dup(219),9 dup(32), 10,13)
   db  11 dup (32),2 dup(219),7 dup(176),2 dup(219),20 dup(32),2 dup(219),7 dup(176),2
dup(219),11 dup(32), 10,13
   db  13 dup (32),2 dup(219),7 dup(176),2 dup(219),16 dup(32),2 dup(219),7 dup(176),2
dup(219),13 dup(32), 10,13
   db  15 dup (32),2 dup(219),7 dup(176),2 dup(219),12 dup(32),2 dup(219),7 dup(176),2
dup(219),15 dup(32), 10,13
   db  17 dup (32),2 dup(219),7 dup(176),2 dup(219),8 dup(32),2 dup(219),7 dup(176),2
dup(219),17 dup(32), 10,13
   db  19 dup (32),26 dup(219),19 dup (32),13,10
   db  60 dup (32),10,13
NumberOfCharsToWrite dd $-Symbol

ReadBuf db 128 dup(?)
hConsoleInput dd 0

.code
start:
call AllocConsole
push -11
call GetStdHandle

mov hConsoleOutput, eax
push 0
push offset NumberOfCharsWritten
push NumberOfCharsToWrite
push offset Symbol
push hConsoleOutput
call WriteConsoleA

push -10
call GetStdHandle
mov hConsoleInput, eax
```

```
push 0
push offset NumberOfCharsWritten
push 128
push offset ReadBuf
push hConsoleInput
call ReadConsoleA

push 0
call ExitProcess
end start
```