

Assembly Language Lab 4

2023/10/16

Objectives

There are two PROC, Convert and Convert2
store a value in esi register then change 0 to A, 1 to B, 2 to C and so on,

Convert PROC **USES** _____ ;Convert change myID value 0-A 1-B and so on.

L1:

 ;Do something

loop L1

ret

Convert ENDP

Convert2 PROC _____ ;Convert2 do the same thing as ConvertL1:

L1:

 ;Do something

loop L1

ret

Convert2 ENDP

Initial State

Registers

EAX = 00009999 EBX = 00009999 ECX = 00010000
EDX = 00009999 ESI = 77ED2938 EDI = 77ECDD00
EIP = 00831057 ESP = 0093F8E8 EBP = 0093F8F4
EFL = 00000206

Memory 1

Address: 0x00834000 Columns: 8

0x00834000	31	30	34	35	32	32	30	32	10452202
0x00834008	31	30	34	35	35	35	31	31	10455511
0x00834010	00	00	00	00	00	00	00	00
0x00834018	00	00	00	00	00	00	00	00
0x00834020	00	00	00	00	00	00	00	00
0x00834028	00	00	00	00	00	00	00	00
0x00834030	00	00	00	00	00	00	00	00

Watch 1

Search (Ctrl+E) Search Depth:

Name	Value
eax	0x00009999
ebx	0x00009999
edx	0x00009999

Add item to watch

- Set **eax**, **ebx** and **edx** values to 9999h at the beginning.
- Make sure the **eax**, **ebx** and **edx** values **do not change** after converting all IDs

Program Result

The screenshot shows a debugger window for 'my_asm_project' at the 'Global Scope'. The assembly code is as follows:

```
37 mov eax, 00000000
38 mov ebx, 00000000
39 mov edx, 00000000
40 mov esi, 00834010
41 mov ecx, 77ECDD00
42 call C
43 mov esi, 00000000
44 mov ecx, 00000000
45 call C
46
47 exit
```

The 'Registers' window shows the following values:

EAX	=	00009999	EBX	=	00009999	ECX	=	00000000
EDX	=	00009999	ESI	=	00834010	EDI	=	77ECDD00
EIP	=	00831075	ESP	=	0093FF0C	EBP	=	0093FF18
EFL	=	00000212						

The 'Memory' window shows the following data:

Address	0x00834000	0x00834008	0x00834010	0x00834018	0x00834020	0x00834028	0x00834030	0x00834038
Hex	42 41 45 46 43 43 41 43	42 41 45 46 46 46 42 42	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
Hex	BAEFCCAC	BAEFFFBF

The 'Output' window shows the following text:

```
Show output from: Debug
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
The thread 0x674c has exited with c
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
'my_asm_project.exe' (Win32): Loaded
```

Hint

The use of push and pop instructions needs to be considered to avoid repeated execution of unnecessary instructions

USES operator: List the temporary registers used in the program, generate a **push** instruction in the beginning of the program, store value of the temporary register into the stack, generate a **pop** instruction reply at the end of the program

Sample Program	The code will be generated by the assembler
<pre>ArraySum PROC USES esi ecx mov eax,0 L1: add eax,[esi] add esi,4 loop L1 ret ArraySum ENDP</pre>	<pre>ArraySum PROC push esi push ecx mov eax,0 L1: add eax,[esi] add esi,4 loop L1 pop ecx pop esi ret ArraySum ENDP</pre>

ASCII CODE

Binary	Decimal	Hexadecimal	Character	Binary	Decimal	Hexadecimal	Character
0011 0000	48	30	0	0100 0001	65	41	A
0011 0001	49	31	1	0100 0010	66	42	B
0011 0010	50	32	2	0100 0011	67	43	C
0011 0011	51	33	3	0100 0100	68	44	D
0011 0100	52	34	4	0100 0101	69	45	E
0011 0101	53	35	5	0100 0110	70	46	F
0011 0110	54	36	6	0100 0111	71	47	G
0011 0111	55	37	7	0100 1000	72	48	H
0011 1000	56	38	8	0100 1001	73	49	I
0011 1001	57	39	9	0100 1010	74	4A	J

Report

- Due to 2023/10/17
- Before 12:00
 - If you submit late, please send a letter to the teaching assistant, points will be deducted
- Group Task
- Compress(.zip,.rar) the following file with the name of the group (e.g. lab4_01.zip)
 - Code(lab4_01.asm)
 - Report(lab4_01.doc or lab4_01.pdf)
 - Report Title
 - Group, name, student ID
 - Step by step of program execution flow, memory (register) status
 - Screenshots description, code Description
 - Reviews for the class, lesson learned, the tools we used, TA, etc