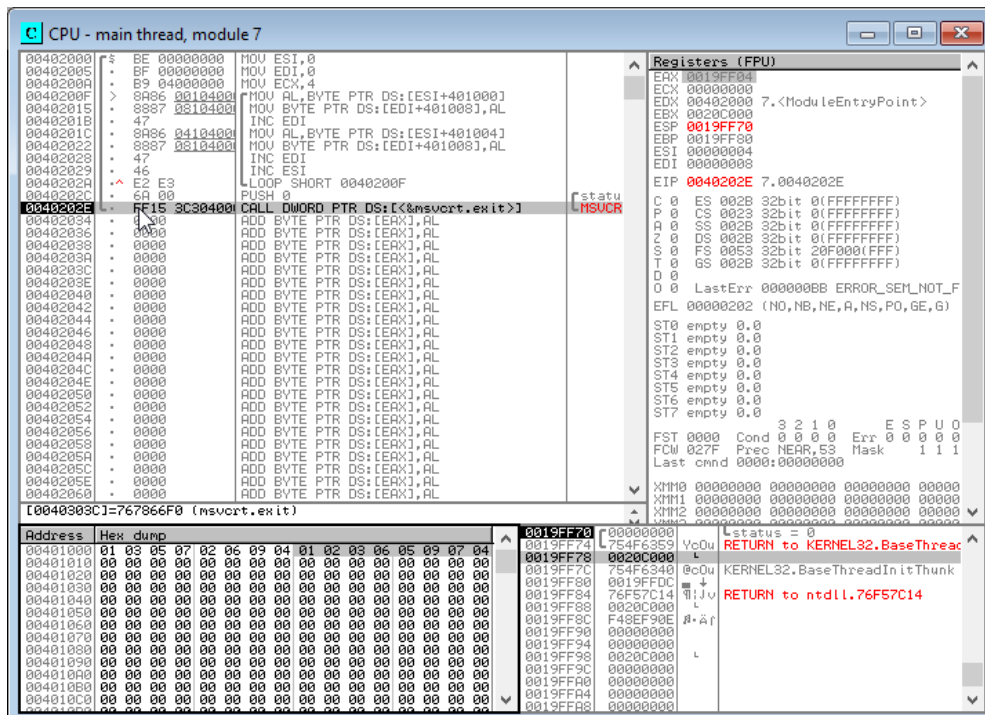


## 7. Se dau două șiruri de octeți $S1$ și $S2$ de aceeași lungime. Să se obțină șirul $D$ prin intercalarea elementelor celor două șiruri.

Data	Code	Iter.1	Iter.2	Iter.3	Iter.4
$S1$ DB 1, 3, 5, 7 ; source 1 $S2$ DB 2, 6, 9, 4 ; source 2 $S\_Len$ EQU \$-S2 ; S is the same for S1 and S2  ; destination $D$ RESB 2 * $S\_Len$	xor ESI, ESI ; ESI = 0 xor EDI, EDI ; EDI = 0 mov ECX, S_Len ; ECX = 4, prepare loop .iterate: mov AL, [S1+ESI] ; AL = S1[ESI] mov [D+EDI], AL ; D[EDI] = AL inc EDI ; EDI++ mov AL, [S2+ESI] ; AL = S2[ESI] mov [D+EDI], AL ; D[EDI] = AL inc EDI ; EDI ++ inc ESI ; ESI ++ loop .iterate ; loop till ECX=0				
	mov AL, [S1+ESI] ; AL = S1[ESI]	AL=1	AL=3	AL=5	AL=7
	mov [D+EDI], AL ; D[EDI] = AL				
	inc EDI ; EDI++	EDI=1	EDI=3	EDI=5	EDI=7
	mov AL, [S2+ESI] ; AL = S2[ESI]	AL=2	AL=6	AL=9	AL=4
	mov [D+EDI], AL ; D[EDI] = AL				
	inc EDI ; EDI ++	EDI=2	EDI=4	EDI=6	EDI=8
	inc ESI ; ESI ++	ESI=1	ESI=2	ESI=3	ESI=4
	loop .iterate ; loop till ECX=0	ECX=3	ECX=2	ECX=1	ECX=0



### Alternativă cu șiruri

```

cld
mov ESI, S1
mov EBX, S_Len-1
mov EDI, D
mov ECX, S_Len
.iterate:
movsb
add ESI, EBX
movsb
stc
sbb ESI, EBX
loop .iterate

```

8. Se dă un șir de caractere *S*. Să se construiască șirul *D* care să conțină toate literele mari din șirul *S*.

Data	Code	I1	I2	I3	I4	I5	I6	I7	I8	Reg
<i>S</i> DB "aAbB2%M"	xor ESI, ESI ; ESI = 0									
<i>D</i> RESB (\$- <i>S</i> )	xor EDI, EDI ; EDI = 0									
	mov ECX, (D- <i>S</i> ) ; ECX = len( <i>S</i> ); prepare loop .iterate:									
	mov AL, [ <i>S</i> +ESI] ; AL = <i>S</i> [ESI]	61h	41h	62h	42h	32h	25h	78h	4Dh	AL
	inc ESI ; ESI++	1	2	3	4	5	6	7	8	ESI
	cmp AL, 'A' ; compare <i>S</i> [ESI] to 'A'	>	=	>	>	<	<	>	>	
	jb .next ; if <i>S</i> [ESI] < 'A', skip	No	No	No	No	Yes	Yes	No	No	
	cmp AL, 'Z' ; compare <i>S</i> [ESI] to 'Z'	>	<	>	<	<	<	>	<	
	ja .next ; if <i>S</i> [ESI] > 'Z', skip	Yes	No	Yes	No	No	No	Yes	No	
	; if we are here, <i>S</i> [ESI] is letter ; so append it to <i>D</i> (estination)	×	↓	×	↓	×	×	×	↓	
	mov [ <i>D</i> +EDI], AL		'A'		'B'				'M'	
	inc EDI ; prepare next dest. index	0	1	1	2	2	2	2	3	EDI
	.next:									
	loop .iterate ; loop until ECX == 0	7	6	5	4	3	2	1	0	ECX
	; end the new string with '\0' character									
	mov [ <i>D</i> +EDI], BYTE 0									

The screenshot shows a debugger window titled "CPU - main thread, module 8". The main window displays assembly code with comments. The code is as follows:

```

00402000 31F6 XOR ESI,ESI
00402002 31FF XOR EDI,EDI
00402004 B9 08000000 MOV ECX,8
00402009 8A86 00104000 MOV AL,BYTE PTR DS:[ESI+401000]
0040200F 46 INC ESI
00402010 3C 41 CMP AL,41
00402012 72 0B JB SHORT 0040201F
00402014 3C 5A CMP AL,5A
00402016 77 07 JA SHORT 0040201F
00402018 8B87 08104000 MOV BYTE PTR DS:[EDI+401008],AL
0040201E 47 INC EDI
0040201F E2 E8 LOOP SHORT 00402009
00402021 C687 08104000 MOV BYTE PTR DS:[EDI+401008],0
00402028 6A 00 PUSH 0
0040202A FF15 3C304000 CALL DWORD PTR DS:[<msvcrt.exit>]
00402030 0000 ADD BYTE PTR DS:[EAX],AL
00402032 0000 ADD BYTE PTR DS:[EAX],AL
00402034 0000 ADD BYTE PTR DS:[EAX],AL
00402036 0000 ADD BYTE PTR DS:[EAX],AL
00402038 0000 ADD BYTE PTR DS:[EAX],AL
0040203A 0000 ADD BYTE PTR DS:[EAX],AL
0040203C 0000 ADD BYTE PTR DS:[EAX],AL
0040203E 0000 ADD BYTE PTR DS:[EAX],AL
00402040 0000 ADD BYTE PTR DS:[EAX],AL
00402042 0000 ADD BYTE PTR DS:[EAX],AL
00402044 0000 ADD BYTE PTR DS:[EAX],AL
00402046 0000 ADD BYTE PTR DS:[EAX],AL
00402048 0000 ADD BYTE PTR DS:[EAX],AL
0040204A 0000 ADD BYTE PTR DS:[EAX],AL
0040204C 0000 ADD BYTE PTR DS:[EAX],AL
0040204E 0000 ADD BYTE PTR DS:[EAX],AL
00402050 0000 ADD BYTE PTR DS:[EAX],AL
00402052 0000 ADD BYTE PTR DS:[EAX],AL
00402054 0000 ADD BYTE PTR DS:[EAX],AL
00402056 0000 ADD BYTE PTR DS:[EAX],AL
00402058 0000 ADD BYTE PTR DS:[EAX],AL

```

The registers window (Registers (FPU)) shows the following values:

- EAX: 0019FF40
- ECX: 00000000
- EDX: 00402000 8.<ModuleEntryPoint>
- EBX: 0026C000
- ESP: 0019FF74 ASCII ""\0\0""
- EBP: 0019FF80
- ESI: 00000008
- EDI: 00000003
- EIP: 00402028 8.00402028

The stack window (Stack [0019FF70]=0) shows the following values:

- Address: 00401000, Hex dump: 61 41 62 42 32 25 78 4D 41 42 4D 00 00 00 00 00
- Address: 00401010, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401020, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401030, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401040, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401050, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401060, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401070, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401080, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 00401090, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 004010A0, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 004010B0, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Address: 004010C0, Hex dump: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00