

Análise do código gerado pelo compilador

- Tabela de alocação de registos

Registos	Variáveis			
%edi	x	x	x	x
%esi				tam
%ebx				y
%ecx	y	y	y	
%edx	dy	dy	dy	
%eax	num	tam	tam	count
	1	2	3	4

Os números 1,2,3 e 4 correspondem a diferentes instantes ao longo da execução da função.

O instante 1 corresponde às instruções antes da verificação da condição `if (lin)`. As instruções da condição `if (lin)` e da condição `else ()` dizem respeito aos instantes 2 e 3, respectivamente. Por fim, as instruções do ciclo `for ()` estão representadas no instante 4.

- Estudo da variável *tab* e indexação da matriz

Sendo a variável *tab* uma matriz 100 por 100 de caracteres, é expectável que ela ocupe cerca de 10 000 bytes. Isto porque a matriz contém cerca de 10 000 elementos (100 linhas x 100 colunas), e cada um desses elementos, sendo um *char*, ocupa 1 byte, perfazendo então o total de 10 000 bytes.

Pela análise do código é possível verificar que esta variável faz parte de uma estrutura. Analisando essa mesma estrutura, constata-se que a variável *tab* é a primeira variável a ser declarada, sendo a variável *lins* a segunda. Como em memória as variáveis da estrutura são armazenadas consecutivamente, uma forma para poder verificar qual o endereço onde está armazenada a variável *tab* passa por verificar onde está armazenada a variável *lins* e, subtrair 10 000 bytes.

Pela análise do código gerado em *Assembly*, a instrução <contar_segs+54> tem por finalidade colocar o valor de *lins* no registo %eax. Desta forma é possível concluir que a variável *lins* está armazenada no endereço de memória dado por %ebp + 0x2718 (Fig.1).

Register group: general							
eax	0x1	1		ecx	0x0	0	
edx	0x0	0		ebx	0x1	1	
esp	0xbfff9bb0	0xbfff9bb0		ebp	0xbfff9bc8	0xbfff9bc8	
esi	0xbfffea48	-1073747384		edi	0x0	0	
eip	0x8048436	0x8048436 <contar_segs+54>		eflags	0x202	[IF]	
cs	0x73	115		ss	0x7b	123	
ds	0x7b	123		es	0x7b	123	
fs	0x0	0		gs	0x33	51	

B+	0x8048406 <contar_segs+6>	sub	\$0xc,%esp
	0x8048409 <contar_segs+9>	xor	%edi,%edi
	0x804840b <contar_segs+11>	xor	%ecx,%ecx
	0x804840d <contar_segs+13>	xor	%edx,%edx
	0x804840f <contar_segs+15>	cmpl	\$0x0,0x2720(%ebp)
	0x8048416 <contar_segs+22>	mov	0x2724(%ebp),%eax
	0x804841c <contar_segs+28>	movl	\$0x0,-0x10(%ebp)
	0x8048423 <contar_segs+35>	movl	\$0x0,-0x14(%ebp)
	0x804842a <contar_segs+42>	je	0x8048486 <contar_segs+134>
	0x804842c <contar_segs+44>	lea	-0x1(%eax),%ecx
	0x804842f <contar_segs+47>	movl	\$0x1,-0x10(%ebp)
	0x8048436 <contar_segs+54>	mov	0x2718(%ebp),%eax
	0x804843c <contar_segs+60>	test	%eax,%eax
	0x804843e <contar_segs+62>	jle	0x804847b <contar_segs+123>

FIGURA 1 - CONTEÚDO DOS REGISTOS APÓS A EXECUÇÃO DA INSTRUÇÃO <CONTAR_SEGS+47>

Pela figura 1 tem-se que o registo %ebp tem o valor 0xbfff9bc8, e somando este com 0x2718 obtém-se o endereço 0xbfffc2e0. A este endereço, subtraindo-se 1000 (0x2710) obtém-se 0xbfff9bd0 que corresponde ao endereço onde está armazenada a variável *tab*.

Uma forma de comprovar este valor passa por verificar qual a posição de memória do primeiro caracter. Na primeira vez que o código é executado, este caracter é passado como argumento para a função <e_seg> assim, analisando a linha <contar_segs+91>, constata-se que o registo %eax contém o endereço do primeiro caracter, fruto da execução da instrução anterior.

Register group: general							
eax	0xbfff9bd0	-1073767472		ecx	0x0	0	
edx	0x0	0		ebx	0x0	0	
esp	0xbfff9bb0	0xbfff9bb0		ebp	0xbfff9bc8	0xbfff9bc8	
esi	0xa	10		edi	0x0	0	
eip	0x804845b	0x804845b <contar_segs+91>		eflags	0x246	[PF ZF IF]	
cs	0x73	115		ss	0x7b	123	
ds	0x7b	123		es	0x7b	123	
fs	0x0	0		gs	0x33	51	

	0x804843c <contar_segs+60>	test	%eax,%eax
	0x804843e <contar_segs+62>	jle	0x804847b <contar_segs+123>
	0x8048440 <contar_segs+64>	mov	%eax,%esi
	0x8048442 <contar_segs+66>	lea	0x0(,%edx,4),%eax
	0x8048449 <contar_segs+73>	add	%edx,%eax
	0x804844b <contar_segs+75>	lea	(%ecx,%ecx,4),%ebx
	0x804844e <contar_segs+78>	mov	%eax,-0x18(%ebp)
	0x8048451 <contar_segs+81>	lea	0x0(%esi),%esi
	0x8048454 <contar_segs+84>	lea	(%ebx,%ebx,4),%eax
	0x8048457 <contar_segs+87>	lea	0x8(%ebp,%eax,4),%eax
	0x804845b <contar_segs+91>	sub	\$0xc,%esp
	0x804845e <contar_segs+94>	movsbl	(%edi,%eax,1),%eax
	0x8048462 <contar_segs+98>	push	%eax
	0x8048463 <contar_segs+99>	call	0x80483e4 <e_seg>

FIGURA 2 - CONTEÚDO DOS REGISTOS APÓS A EXECUÇÃO DA INSTRUÇÃO <CONTAR_SEGS+87>

Pela figura 2, comprova-se que o endereço %eax contém o valor previamente calculado.

Posto isto, tem-se que esta zona de memória se encontra organizada da forma representada na Tabela 1:

Endereços Memória	
0xbff9bd0	tab [1][1]
0xbff9bd1	tab [1][2]
...	...
...	tab[1][100]
...	tab[2][1]
...	...
0xbffc2df	tab[100][100]
0xbffc2e0	lins

TABELA 1 - ORGANIZAÇÃO EM MEMÓRIA DA VARIÁVEL *TAB*

A matriz é armazenada em memória linha a linha. Assim, a forma de aceder a um elemento da matriz é feito do seguinte modo:

$\text{endereço_elemento} = (\text{linha_elemento}) \cdot (\text{coluna_elemento}) \cdot (\text{n}^\circ_elementos_por_linha) + (\text{endereço_da_matriz}).$

As linhas de código *Assembly* responsáveis por fazer a indexação da matriz são então:

```
<contar_segs+64>:  mov  %eax,%esi
<contar_segs+66>:  lea   0x0(,%edx,4),%eax
<contar_segs+73>:  add   %edx,%eax
<contar_segs+75>:  lea   (%ecx,%ecx,4),%ebx
<contar_segs+78>:  mov   %eax,-0x18(%ebp)
<contar_segs+81>:  lea   0x0(%esi),%esi
<contar_segs+84>:  lea   (%ebx,%ebx,4),%eax
<contar_segs+87>:  lea   0x8(%ebp,%eax,4),%eax
<contar_segs+91>:  sub   $0xc,%esp
<contar_segs+94>:  movsbl (%edi,%eax,1),%eax
<contar_segs+91>:  sub   $0xc,%esp
<contar_segs+94>:  movsbl (%edi,%eax,1),%eax
```