

# IA32 : Modos de Endereçamento

## Guião IV - Resolução

Questão 1 (*cálculo de endereços*):

a)

Operando	Valor	Comentário
%eax	0x100	Registo
0x104	0xAB	Endereço Absoluto
\$0x108	0x108	Imediato
(%eax)	0xFF	Endereço 0x100
4(%eax)	0xAB	Endereço 0x104
9(%eax,%edx)	0x11	Endereço 0x10C
260(%ecx,%edx)	0x13	Endereço 0x108
0xFC(,%ecx,4)	0xFF	Endereço 0x100
(%eax,%edx,4)	0x11	Endereço 0x10C

b)

Instrução	Destino	Valor	Z	S	C	O
addl %ecx, (%eax)	Mem[0x100 : 0x103]	0x100	0	0	0	0
subl %edx, 4(%eax)	Mem[0x104 : 0x107]	0xA8	0	0	0	0
imull \$16, (%eax,%edx,4)	Mem[0x10C : 0x10F]	0x110	0	0	0	0
incl 8(%eax)	Mem[0x108 : 0x10B]	0x14	0	0	0	0
decl %ecx	%ecx	0x0	1	0	0	0
addl \$0x7fffffff, %edx	%edx	0x80000001	0	1	0	0
addl \$-1, %eax	%eax	0xFF	0	0	0	0
andl \$-4, (%eax)	Mem[0x100 : 0x103]	0xFC	0	0	0	0
cmpl %eax, %edx	cc <- (0x03 - 0x100)	-	0	1	1	0
test 0x08(%eax), %edx	cc <- (0x13 & 0x0x3)	-	0	0	0	0
addb 0x7F, %cl (**)	%cl	0x80	0	1	0	1
subl %edx, %eax	%eax	0xFD	0	0	0	0

(\*\*) Exemplo complementar para mostrar CC ← transbordo (*overflow*)

## Questão 2 (*cálculo de expressões*):

Instrução	Valor
leal 6(%eax), %edx	$x + 6$
leal (%eax,%ecx), %edx	$x + y$
leal (%eax,%ecx,4), %edx	$x + 4*y$
leal 7(%eax,%eax,8), %edx	$9*x + 7$
leal 0xA(,%ecx,4), %edx	$4*y + 10$
leal 9(%eax,%ecx,2), %edx	$x + 2*y + 9$
leal 0xFFFFFDE(%eax,%ecx,4), %edx	$x + 4*y - 34$

## Questão 3 (*execução e depuração de programas*)

Alguns comandos do depurador GDB úteis à realização deste exercício:

```
(gdb) print x          -> mostra o conteúdo da variável x
(gdb) info address x    -> mostra informação (incluindo o endereço) da variável x
(gdb) print &x          -> mostra o endereço da variável x
(gdb) print *0x804963c  -> mostra o conteúdo do endereço 0x804963c
(gdb) info all-registers -> mostra o conteúdo de todos os registos
(gdb) print $ebp        -> mostra o conteúdo do registo %ebp
(gdb) info register ebp  -> mostra o conteúdo do registo %ebp
```

a)

**main():**

```
0x08048354 <main+0>: push    %ebp
0x08048355 <main+1>: mov     %esp,%ebp
0x08048357 <main+3>: sub     $0x8,%esp
0x0804835a <main+6>: and     $0xffffffff0,%esp
0x0804835d <main+9>: mov     $0x0,%eax
0x08048362 <main+14>: sub     %eax,%esp
0x08048364 <main+16>: call    0x8048370 <movBits>
0x08048369 <main+21>: mov     $0x0,%eax
0x0804836e <main+26>: leave
0x0804836f <main+27>: ret
```

**movBits():**

```
0x08048370 <movBits+0>: push    %ebp
0x08048371 <movBits+1>: mov     %esp,%ebp
0x08048373 <movBits+3>: sub     $0x4,%esp
0x08048376 <movBits+6>: movl    $0xffffffff,-0x4(%ebp)    ; t = -1
0x0804837d <movBits+13>: shll    $0x3,0x804962c            ; x = x << 3
0x08048384 <movBits+20>: mov     0x804962c,%eax
0x08048389 <movBits+25>: mov     %eax,0x804963c            ; u = x
0x0804838e <movBits+30>: mov     0x804962c,%ax
0x08048394 <movBits+36>: mov     %ax,0x8049638            ; s = x
0x0804839a <movBits+42>: mov     0x804962c,%al
0x0804839f <movBits+47>: mov     %al,0x8049640            ; c = x
0x080483a4 <movBits+52>: shll    $0x3,0x804963c            ; u = u << 3
0x080483ab <movBits+59>: mov     -0x4(%ebp),%eax
0x080483ae <movBits+62>: mov     %eax,0x804962c            ; x = t
0x080483b3 <movBits+67>: mov     0x8049630,%cl
0x080483b9 <movBits+73>: sarl    %cl,0x804962c            ; x = x >> n
0x080483bf <movBits+79>: mov     0x804962c,%eax
0x080483c4 <movBits+84>: mov     %eax,0x804963c            ; u = x
0x080483c9 <movBits+89>: mov     0x804962c,%ax
0x080483cf <movBits+95>: mov     %ax,0x8049638            ; s = x
0x080483d5 <movBits+101>: mov     0x804962c,%al
0x080483da <movBits+106>: mov     %al,0x8049640            ; c = x
0x080483df <movBits+111>: mov     0x8049630,%cl
0x080483e5 <movBits+117>: shrll   %cl,0x804963c            ; u = u >> n
0x080483eb <movBits+123>: leave
0x080483ec <movBits+124>: ret
```

Variável	Endereço
<b>t</b>	EBP-4 = 0xbfffe948-4 = 0xbfffe944
<b>x</b>	0x804962c
<b>n</b>	0x8049630
<b>s</b>	0x8049638
<b>u</b>	0x804963c
<b>c</b>	0x8049640

b) No início **x=128** e **n=31**

Instrução C	Valor da variável	Valores dos registos
t=-1	t=-1	EBP=bfffe948
x <= 3	x=1024	
u=x	u=1024	EAX=1024 (0x0000 <b>0400</b> )
s=x	s=1024	<b>AX</b> =1024 ( 0x04 <b>00</b> )
c=x	c=0	<b>AL</b> =0 ( 0x00)
u <= 3	u=8192	
x=t	x=-1	EAX=-1, EBP=bfffe948
x >= n	x=-1 (*)	CL =0x1F=31
u=x	u=4294967295	EAX=-1 (0xFFFF <b>FFFF</b> )
s=x	s=0xFFFF=-1	<b>AX</b> =-1 ( 0xFF <b>FF</b> )
c=x	c=0xFF=-1	<b>AL</b> =-1 ( 0xFF)
u >= n	u=1 (**)	CL =0x1F=31

(\*\*) dá um resultado diferente de (\*) porque **x** é **int** (o deslocamento à direita ">>31" é implementado com um **sarl**), enquanto **u** é **unsigned int** (o deslocamento à direita ">>31" é implementado com um **shrl**).