Instruction set do IA-32

Tipo	Instrução	Efeito	Descrição
Transferência de Informação	mov? S, D	D←S	Move (? = b,w,l)
	movsbl S, D	D←SignExtend(S)	Move Sign-Extended Byte
	movzbl S, D	D←ZeroExtend(S)	Move Zero-Extended Byte
	pushl S	%esp ← %esp - 4; Mem[%esp] ← S	Push
	popl D	D←Mem[%esp]; %esp ←%esp+ 4	Pop
	leal S, D	D← &S	Load Effective Address
Operações Aritméticas e Lógicas	incl D	D← D +1	Increment
	decl D	D← D –1	Decrement
	negl D	D← -D	Negate
	notl D	D← ~D	Complement
	addl S, D	D← D + S	Add
	subl S, D	D← D – S	Subtract
	imull S, D	D← D * S	32 bit Multiply
	xorl S, D	D← D ^ S	Exclusive-Or
	orl S, D	D← D | S	Or
	andl S, D	D← D & S	And
	sall k, D	D← D << k	Left Shift
	shll k, D	D← D << k	Left Shift
	sarl k, D	D← D >> k	Arithmetic Right Shift
	shrl k, D	D← D >> k	Logical Right Shift
	imull S	%edx : %eax ← S × %eax	Signed 64 bit Multiply
	mull S	%edx : %eax ← S × %eax	Unsigned 64 bit Multiply
	cltd	%edx : %eax ← SignExtend(%eax)	Convert to Quad Word
	idivl S	%edx ← %edx : %eax mod S; %eax ← %edx:%eax ÷ S	Signed Divide
	divl S	%edx ← %edx : %eax mod S; %eax ← %edx : %eax ÷ S	Unsigned Divide
Teste	cmp? S2, S1	(CF, ZF, SF, OF) ← S1 – S2	Compare (? = b,w,l)
	test? S2, S1	(CF, ZF, SF, OF) ← S1 & S2	Test (? = b,w,l)
Instruções de set	sete R8	R_8 ← ZF (Sinónimo: setz R8)	Equal/Zero
	setne R8	R_8 ← ~ZF (Sinónimo: setnz R8)	Not Equal/Not Zero
	sets R8	R_8 ← SF	Negative
	setns R8	R_8 ← ~SF	Non Negative
	setg R8	R_8 ← ~(SF^OF) & ~ZF (Sinónimo: setnle R8)	Greater (signed >)
	setge R8	R_8 ← ~(SF^OF) (Sinónimo: setnl R8)	Greater or equal (signed >=)
	setl R8	R_8 ← SF^OF (Sinónimo: setnge R8)	Less (signed <)
	setle R8	R_8 ← (SF^OF) | ZF (Sinónimo: setng R8)	Less or equal (signed <=)
	seta R8	R_8 ← ~CF & ~ZF (Sinónimo: setnbe R8)	Above (unsigned >)
	setae R8	R_8 ← ~CF (Sinónimo: setnb R8)	Above or equal (unsigned >=)
	setb R8	R_8 ← CF (Sinónimo: setnae R8)	Below (unsigned <)
	setbe R8	R_8 ← CF & ~ZF (Sinónimo: setna R8)	Below or equal (unsigned <=)
Instruções de salto	jmp Label	%eip ← Label	Unconditional jump
	jmp *D	%eip ← *D	Indirect unconditional jump
	je Label	Jump if ZF (Sinónimo: jz)	Zero/Equal
	jne Label	Jump if ~ZF (Sinónimo: jnz)	Not Zero/Not Equal
	js Label	Jump if SF	Negative
	jns Label	Jump if ~SF	Not Negative
	jg Label	Jump if ~(SF ^OF) & ~ZF (Sinónimo: jnle)	Greater (signed >)
	jge Label	Jump if ~(SF^OF) (Sinónimo: jnl)	Greater or equal (signed >=)
	jl Label	Jump if SF^OF (Sinónimo: jnge)	Less (signed <)
	jle Label	Jump if (SF^OF) | ZF (Sinónimo: jng)	Less or equal (signed <=)
	ja Label	Jump if ~CF & ~ZF (Sinónimo: jnbe)	Above (unsigned >)
	jae Label	Jump if ~CF (Sinónimo: jnb)	Above or equal (unsigned >=)
	jb Label	Jump if CF (Sinónimo: jnae)	Below (unsigned <)
	jbe Label	Jump if CF & ~ZF (Sinónimo: jna)	Below or equal (unsigned <=)
Invocação de Procedimentos	call Label	pushl %eip; %eip= Label	Procedure call
	call *Op	pushl %eip; %eip= *Op	Procedure call
	ret	popl %eip	Procedure return
	leave	movl %ebp, %esp; pop %ebp	Prepare stack for return

D – destino [Reg | Mem] S – fonte [Imm | Reg | Mem] R_8 – destino Reg 8 *bits*

D e S não podem ser ambos operandos em memória

Alberto Proença & Luís Paulo Santos, Mar-03

```
int M2m (char *s) {
  int i=0, count=0;

  while (s[i] != '\0') {
    if (s[i]>='A' && s[i]<='Z') {
      count++;
      s[i] += 32; }
    i++;  }
  return count; }

int main () {
  char str[50];
  int c;

  scanf ("%s", str);
  c = M2m (str);
  printf ("%s\n(Converted %d)\n", str, c);
}
```

---- código assembly obtido com gcc -S -O0 ----

```
...
M2m:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        movl    $0, -4(%ebp)
        movl    $0, -8(%ebp)
.L2:
        movl    -4(%ebp), %eax
        addl    8(%ebp), %eax
        cmpb    $0, (%eax)
        jne     .L4
        jmp     .L3
.L4:
        movl    -4(%ebp), %eax
        addl    8(%ebp), %eax
        cmpb    $64, (%eax)
        jle     .L5
        movl    -4(%ebp), %eax
        addl    8(%ebp), %eax
        cmpb    $90, (%eax)
        jg      .L5
        leal    -8(%ebp), %eax
        incl    (%eax)
        movl    -4(%ebp), %eax
        movl    8(%ebp), %edx
        addl    %eax, %edx
        movl    -4(%ebp), %eax
        addl    8(%ebp), %eax
        movb    (%eax), %al
        addl    $32, %eax
        movb    %al, (%edx)
.L5:
        leal    -4(%ebp), %eax
        incl    (%eax)
        jmp     .L2
.L3:
        movl    -8(%ebp), %eax
        leave
        ret
```

---- código assembly obtido com gcc -S -O2 ----

```
...
M2m:
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %esi
        pushl   %ebx
        movl    8(%ebp), %ebx
        movb    (%ebx), %al
        xorl    %ecx, %ecx
        xorl    %esi, %esi
        testb   %al, %al
        je      .L8
        movb    %al, %dl
.L6:
        leal    -65(%edx), %eax
        cmpb    $25, %al
        ja      .L5
        leal    32(%edx), %eax
        incl    %esi
        movb    %al, (%ecx,%ebx)
.L5:
        incl    %ecx
        movb    (%ecx,%ebx), %al
        testb   %al, %al
        movb    %al, %dl
        jne     .L6
.L8:
        popl    %ebx
        movl    %esi, %eax
        popl    %esi
        leave
        ret
```

---- executável com -O2 após objdump -d -----

```
...
080483c4 <M2m>:
 80483c4:  55              push  %ebp
 80483c5:  89 e5           mov  %esp,%ebp
 80483c7:  56              push %esi
 80483c8:  53              push %ebx
 80483c9:  8b 5d 08        mov 0x8(%ebp),%ebx
 80483cc:  8a 03           mov (%ebx),%al
 80483ce:  31 c9           xor %ecx,%ecx
 80483d0:  31 f6           xor %esi,%esi
 80483d2:  84 c0           test %al,%al
 80483d4:  74 1a           je 80483f0 <M2m+0x2c>
 80483d6:  88 c2           mov %al,%dl
 80483d8:  8d 42 bf        lea 0xffbf(%edx),%eax
 80483db:  3c 19           cmp $0x19,%al
 80483dd:  77 07           ja 80483e6 <M2m+0x22>
 80483df:  8d 42 20        lea 0x20(%edx),%eax
 80483e2:  46              inc %esi
 80483e3:  88 04 19        mov %al,(%ecx,%ebx,1)
 80483e6:  41              inc %ecx
 80483e7:  8a 04 19        mov (%ecx,%ebx,1),%al
 80483ea:  84 c0           test %al,%al
 80483ec:  88 c2           mov %al,%dl
 80483ee:  75 ??           jne 80483d8 <M2m+0x14>
 80483f0:  5b              pop %ebx
 80483f1:  89 f0           mov %esi,%eax
 80483f3:  5e              pop %esi
 80483f4:  c9              leave
 80483f5:  c3              ret
```

- breakpoint na f. M2m (executável com -O2) _

```
(gdb) info registers
eax            0x6e       110
ecx            0x4        4
edx            0x6e       110
ebx            0xbfffe870 -1073747856
esp            0xbfffe850 0xbfffe850
ebp            0xbfffe858 0xbfffe858
esi            0x2        2
edi            0x0        0
eip            0x80483d8  0x80483d8 <M2m+20>
eflags         0x293      [ IF ]
```

```
 (gdb) x/32xb $esp
0xbfffe850:
   0x70   0xe8  0xff  0xbf  0xa0  0x3c  0x57  0x00
0xbfffe858:
   0xb8   0xe8  0xff  0xbf  0x1b  0x84  0x04  0x08
0xbfffe860:
   0x70   0xe8  0xff  0xbf  0x70  0xe8  0xff  0xbf
0xbfffe868:
   0x00   0x00  0x00  0x00  0x40  0x83  0x04  0x08
```