

**Ex. 1** Para cada uma das estruturas de dados mostradas abaixo, compute seu `sizeof`, e compute o endereço de byte do elemento indicado no comentário.

```

1 int a; // sizeof( a ) ?
2 int x[NX], y[NY]; // sizeof( x ) ?
3 short z[NZ]; // sizeof( z ) ?
4 char w[NW]; // sizeof( w ) ?
5 ...
6 a=x[ y[ 3 ] ]; // sizeof( y[5] ) ? endereço de x[y[5]] ?
7 ...
8 typedef elemType { // sizeof( elemType ) ?
9     elemType *next; // sizeof( next ) ?
10    int vet[16]; // sizeof( vet ) ?
11 } elemType;
12 elemType strut[256]; // endereço de strut[5].vet[5] ?
13
14 typedef struct A {
15     int x; // sizeof( x ) ?
16     short s[4]; // sizeof( s ) ?
17     char c[4]; // sizeof( c ) ?
18 } aType; // sizeof( aType ) ?
19 aType V[1024]; // endereço de V[13].c[2] ?

```

**Ex. 2** Mostre o código MIPS necessário para implementar os seguintes comandos em C.

```

1 // (a)
2 int a,b,i;
3 int x[NNN], y[MMM];
4 ...
5 a = x[i] + x[ y[3] ];
6 i = y[i+1]/4;
7 b = x[i] + y[ x[2*i] ];
8
9 // (b)
10 typedef elem {
11     elem *next;
12     int vet[6];
13 } elemType;
14
15 elemType *p;
16 elemType strut[256];
17 ...
18 p = strut; // p <- &(strut[0])
19 if (p->next != NULL) {
20     p->vet[0] = 1;
21     p->vet[2] = 1024;
22     p = p->next;
23 }
24 ...

```

**Ex. 3** Traduza para *assembly* do MIPS os trechos de programa em C, listados abaixo.

```

1 // (a)
2 int i=1, x=0; int Z[100];
3 ...
4 do {
5     x = x + Z[i];
6     i = i * 4;
7 } until (i < 100);
8
9 // (b)
10 int fun(int a, int b, int c, int d, int e, int f);
11 ...
12     x = fun(16*a, z*w, p^q, (z=v[3]), v[3*z], z-2);
13
14 // (c)
15 typedef struct A {
16     int x;
17     short s[4];
18     char c[8];
19 } aType;
20 aType V[1024]; // compilador aloca V em 0x0080.0000
21 int i,a,b,c;
22 ...
23 for (a=b=c=i=0; i < 1024; i++) {
24     a = a + V[i].x + (int)V[i].s[1];
25     b = b + (int)(V[i].c[5] + V[i].c[7]);
26     c = c + (int)V[i].s[2] - (int)V[i].s[3];
27 }
28
29 // (d)
30 int log2(int n) {
31     if (n < 2) then
32         return 0;
33     else
34         return (1 + log2(n/2)); # divisão com deslocamento
35 }
36 ...
37     x = log2(96000);
38 ...
39
40 // (e)
41 typedef elem {
42     elem *next;
43     int vet[3];
44 } elemType;
45
46 elemType *x;
47 elemType strut[256];
48 ...
49     x = insert( strut, j) );
50     x->vet[2] = 512;
51 ...
52 elemType * insert(elemType *p, int i) {
53
54     while (p != NULL) {
55         p = p->next;
56     }
57     p->next = &(strut[i]);
58     (p->next)->next = NULL;
59     return p->next;
60 }

```