O'Hallaron: CS:APP, 2ª Ed.

Signatura: ESIIT/C.1 BRY com

Capítulo 3: Representación de Programas a nivel de máquina

Problemas Prácticos T2.5:

3.40-3.42, pp.281,285

3.53, p.325

3.40. Suponer que nos asignan la tarea de comprobar que un compilador de C genera correctamente código para acceder a estructuras y uniones. Escribimos la siguiente declaración de estructura:

```
typedef union {
    struct {
        short    v;
        short    d;
        int      s;
    } t1;
    struct {
        int a[2];
        char  *p;
    } t2;
} u_type;
```

Escribimos una serie de funciones de la forma

```
void get(u_type *up, TYPE *dest) {
    *dest = EXPR;
}
```

con diferentes expresiones de acceso EXPR, y con tipo de datos destino TYPE ajustado de acuerdo con el tipo asociado a EXPR. Examinamos entonces el código generado al compilar las funciones para ver si concuerdan con nuestras expectativas. Suponer que en estas funciones `up` y `dest` están cargados en los registros `%eax` y `%edx`, respectivamente. Rellenar la siguiente tabla con el tipo de datos TYPE y las secuencias de 1-3 instrucciones para calcular la expresión y almacenar el resultado en `dest`. Intentar usar sólo los registros `%eax` y `%edx`, usando el registro `%ecx` cuando no baste con los dos.

EXPR	TYPE	Código
`up->t1.s`	`int`	`movl 4(%eax), %eax` `movl %eax, (%edx)`
`up->t1.v`	_____	_____

		_____
&up->t1.d	_____	_____
up->t2.a	_____	_____
up->t2.a[up->t1.s]	_____	_____
*up->t2.p	_____	_____

3.41. Para cada una de las siguientes declaraciones de estructuras, determinar el desplazamiento de cada campo, el tamaño total de la estructura, y sus requisitos de alineamiento bajo Linux/IA32.

A. `struct P1 { int i; char c; int j; char d; };`
B. `struct P2 { int i; char c; char d; int j; };`
C. `struct P3 { short w[3]; char c[3]; };`
D. `struct P4 { short w[3]; char *c[3]; };`
E. `struct P5 { struct P1 a[2]; struct P2 *p };`

3.42. Para la declaración de estructura

```
struct {
    char      *a;
    short      b;
    double     c;
    char       d;
    float      e;
    char       f;
    long long  g;
    void      *h;
} foo;
```

suponer que fue compilada en una máquina Windows, donde cada tipo de datos primitivo de *K* bytes debe tener un desplazamiento múltiplo de *K*.

A. ¿Cuáles son los tamaños y desplazamientos en bytes de todos los campos de la estructura?
B. ¿Cuál es el tamaño total de la estructura?
C. Reordenar los campos de la estructura para minimizar el espacio desperdiciado, y entonces mostrar los desplazamientos en bytes y tamaño total para la estructura reordenada.

3.53. Para cada una de las siguientes declaraciones de estructuras, determinar el desplazamiento de cada campo, el tamaño total de la estructura, y sus requisitos de alineamiento bajo x86-64.

F. `struct P1 { int i; char c; long j; char d; };`
G. `struct P2 { long i; char c; char d; int j; };`
H. `struct P3 { short w[3]; char c[3]; };`
I. `struct P4 { short w[3]; char *c[3]; };`
J. `struct P5 { struct P1 a[2]; struct P2 *p };`