O'Hallaron: CS:APP, 2ª Ed.

Signatura: ESIIT/C.1 BRY com

Capítulo 3: Representación de Programas a nivel de máquina

Problemas Prácticos:

3.1-3.5, pp.204,208,210-11

3.30, p.257

3.46-3.47, pp.305,310

3.1. Asumir que los siguientes valores están almacenados en los registros y direcciones de memoria que se indican:

Dirección	Valor
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

Registro	Valor
%eax	0x100
%ecx	0x1
%edx	0x3

Rellenar la siguiente tabla mostrando los valores de los operandos indicados:

Operando	Valor
%eax	
0x104	
$0x108	
(%eax)	
4(%eax)	

Operando	Valor
9(%eax,%edx)	
260(%ecx,%edx)	
0xFC(,%ecx,4)	
(%eax,%edx,4)	

3.2. Para cada una de las siguientes líneas de lenguaje ensamblador, determinar el sufijo de instrucción apropiado basándose en los operandos (por ejemplo, `mov` puede rescribirse como `movb`, `movw`, o `movl`).

1	`mov %eax, (%esp)`
2	`mov (%eax), %dx`
3	`mov $0xFF, %bl`
4	`mov (%esp,%edx,4), %dh`
5	`push $0xFF`
6	`mov %dx, (%eax)`
7	`pop %edi`

3.3. Cada una de las siguientes líneas de código genera un mensaje de error cuando la pasamos por el ensamblador. Explicar qué está mal en cada línea.

1	`movb $0xF, (%bl)`
2	`movl %ax, (%esp)`
3	`movw (%eax), 4(%esp)`
4	`movb %ah, %sh`

```
5         movl %eax, $0x123
6         movl %eax, %dx
7         movb %si, 8(%ebp)
```

3.4. Asumir las variables v y p declaradas con los tipos:

```
src_t v;
dest_t *p;
```

donde `src_t` y `dest_t` son tipos de datos declarados con `typedef`. Deseamos usar la instrucción de transferencia apropiada para implementar la operación

```
*p = (dest_t) v;
```

donde v está almacenada en la parte del registro `%eax` de nombre apropiado al tamaño (esto es, `%eax`, `%ax`, o `%al`), mientras que el puntero p está almacenado en el registro `%edx`.
Para las siguientes combinaciones de `src_t` y `dest_t`, escribir una línea de código ensamblador que haga la transferencia apropiada. Recordar que cuando se realiza un cambio de tipo en lenguaje C que implica ambas cosas, cambio de tamaño y de interpretación del signo, la operación debería cambiar primero la interpretación de signo (Sección 2.2.6, se refiere a que se extiende con la interpretación del tipo fuente, y ya se verá, cuando se consulte el valor destino con su interpretación de signo, qué valor saldrá; la solución puede comprobarse escribiendo el correspondiente programa en C y desensamblándolo).

src_t	dest_t	Instrucción
int	int	movl %eax, (%edx)
char	int	
char	unsigned	
unsigned char	int	
int	char	
unsigned	unsigned char	
unsigned	int	

3.5. Se nos da la siguiente información. Una función con prototipo

```
void decode1(int *xp, int *yp, int *zp)
```

se compila a código ensamblador. El cuerpo del código resultante es como sigue:

```
  xp en %ebp+8, yp en %ebp+12, zp en %ebp+16
1         movl 8(%ebp), %edi
2         movl 12(%ebp), %edx
3         movl 16(%ebp), %ecx
4         movl (%edx), %ebx
5         movl (%ecx), %esi
6         movl (%edi), %eax
7         movl %eax, (%edx)
8         movl %ebx, (%ecx)
9         movl %esi, (%edi)
```

Los parámetros `xp`, `yp` y `zp` están almacenados en posiciones de memoria con desplazamientos 8, 12 y 16, respectivamente, relativos a la dirección en el registro `%ebp`.
Escribir código C para `decode1` que tenga un efecto equivalente al código ensamblador mostrado arriba.

3.30. El siguiente fragmento de código se presenta con frecuencia en la versión compilada de rutinas de librería:

```
1        call next
2   next:
3        popl %eax
```

A. ¿Qué valor resulta asignado al registro `%eax`?
B. Explicar por qué no hay una instrucción `ret` correspondiente a esta `call`.
C. ¿Qué propósito o utilidad tiene este fragmento de código?

3.46. Como se muestra en la Figura 6.17(b), el coste de la DRAM, la tecnología de memoria utilizada para implementar la memoria principal de los microprocesadores, ha caído desde alrededor de $8,000 por megabyte en 1980 hasta alrededor de $0.06 en 2010, aproximadamente un factor de 1.48 cada año, o alrededor de 51 cada 10 años. Asumamos que estas tendencias continuarán indefinidamente (lo cual puede no ser realista), y que nuestro presupuesto para la memoria de una máquina es alrededor de $1,000, de manera que hubiéramos configurado una máquina con 128kilobytes en 1980 y con 16.3 gigabytes en 2010.

A. Estimar cuándo nuestro presupuesto de $1,000 podrá comprar 256 terabytes de memoria.
B. Estimar cuándo nuestro presupuesto de $1,000 podrá comprar 16 exabytes de memoria.
C. ¿Cuánto más temprano sucederían estos puntos de transición si subiéramos nuestro presupuesto para DRAM a $10,000? (Pista: calcular cuántos años se ahorran por incrementar el presupuesto x10 y aplicarlo a las soluciones anteriores)
D. Preguntas añadidas: indicar cómo se han calculado en el enunciado las cantidades 1.48/año, 51/10años, 128KB, 16.3GB

Figura 6.17(b)

Métrica	1980	1985	1990	1995	2000	2005	2010	2010:1980
$/MB	8000	880	100	30	1	.1	0.06	130,000
Tacc(ns)	375	200	100	70	60	50	40	9
Tam.típ.(MB)	0.064	0.256	4	16	64	2,000	8,000	125,000

3.47. La siguiente función C convierte un argumento de tipo `src_t` a un valor de retorno de tipo `dst_t`, donde estos dos tipos se definen usando `typedef`:

```
dest_t cvt(src_t x)
{
    dest_t y = (dest_t) x;
    return y;
}
```

Asumir que el argumento x está en la parte del registro `%rdi` de nombre apropiado al tamaño (esto es, `%rdi`, `%edi`, `%di`, o `%dil`), y que algún tipo de instrucción de transferencia ha de ser usada para realizar la conversión de tipo y copiar el valor a la parte del registro %rax de nombre apropiado. Rellenar la siguiente tabla indicando las instrucciones, el registro fuente, y el registro destino para las siguientes combinaciones de tipo fuente y destino:

src_t	dest_t	Instrucción	S	D
long	long	movq	%rdi	%rax
int	long			
char	long			
unsigned int	unsigned long			
unsigned char	unsigned long			
long	int			
unsigned long	unsigned			