Notas de la Clase

Luis Eduardo Galindo Amaya

27-09-2022

Índice

1. Trayectoria de datos **4**

2. Organización de un sistema computarizado **4**

3. Funciones de una computadora **5**

4. Diagrama de bloques de una computadora **5**

5. Unidad Central de Procesamiento (CPU) **6**
5.1. Componentes de CPU 6
5.2. Funciones principales del CPU 6

6. Jerarquía de Memoria **7**

7. Paginación de memoria (Probablemente no venga) **7**

8. Arquitectura de la computadora **8**
8.1. Sección de Entrada/Salida 8
8.2. Bus de direcciones 8
8.3. Bus de datos 8
8.4. Bus de Control 9

9. Diferencias de CISC con RISC **9**
9.1. CISC 9
9.2. RISC 9
9.3. Tabla de diferencias 10

10. Arquitectura de Von Neuman **11**

11. Arquitectura de Harvard **11**

12. Taxonomía de Flynn **12**

13. Sistema de numeración binario **12**
13.1. Conversión de binario a decimal 12
13.2. Conversión decimal a binario 12
13.3. Suma binaria 12
13.4. Multiplicación binaria 13
13.5. División entre números binarios 13
13.6. Números negativos en binario 13
13.7. Representación decimal a binario 15
13.8. Binario con punto decimal a flotante 16
13.9. BCD (Binary Coded Decimal) 16

14. Sistema Hexadecimal **16**
14.1. Ventajas 16
14.2. Conversión decimal a hexadecimal 17
14.3. Conversión hexadecimal a decimal 17
14.4. Conversión binario a hexadecimal 17

15. Números flotantes **18**
15.1. Distribución de bits de un numero flotante 18
15.2. Convertir decimal a binario 18
15.3. Convertir representación binario a decimal 20

16. ¿Que es un traductor? **21**

17. Traducción vs Interpretación **21**

18. Lenguajes de alto y bajo nivel **21**

19. Estructura de una instruccion de ensamblador **22**
19.1. Estructura 22

20. Tipos de ensambladores **22**
20.1. de dos pasos 22
20.2. de un paso 23

21. Linking (Enlazamiento) **23**
21.1. ¿Que es? 23
21.2. ¿Como lo hace? 23
21.3. ¿para que lo hace? 24

1. Trayectoria de datos

Camino de los datos en una máquina von Neumann típica. Los registros almacenan datos temporalmente. Los registros son usados como entradas de la ALU. La ALU genera una operación sobre los registros. El resultado de la operación vuelve a ser almacenado en los registros.

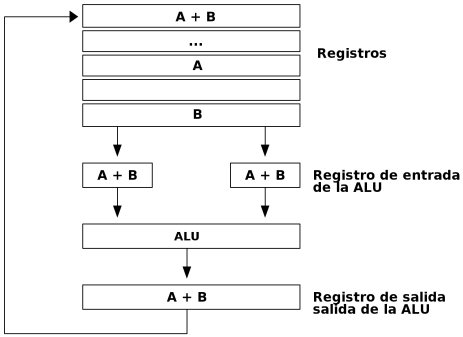


Figura 1: Camino de datos en una arquitectura de Von neuman.

2. Organización de un sistema computarizado

- Las bases que conforman un sistema computarizado son:
  - La Unidad Central de Procesamiento (CPU)
  - La sección de Memoria
  - La sección de Entrada/Salida (E/S)
- Estas tres secciones están interconectadas por tres conjuntos de líneas paralelas llamadas buses o ductos.
- Estos buses son:
  - Bus de Direcciones

- Bus de Datos
- Bus de Control

3. Funciones de una computadora

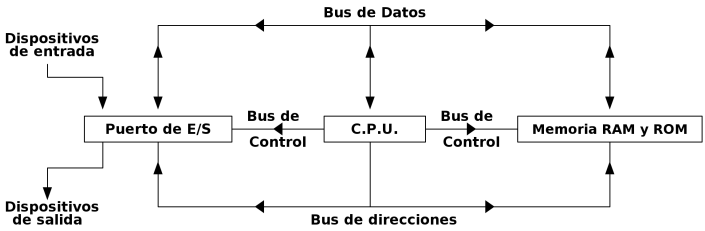
Procesamiento de datos Los datos pueden adoptar una gran variedad de formas, y el rango de los requisitos de procesado es amplio.

Almacenamiento de datos La computadora tiene que guardar temporalmente al menos aquellos datos con los que está trabajando en un momento dado.

Transferencia de datos El entorno de operación del computador se compone de dispositivos que sirven bien como fuente o bien como destino de datos.

Control Dentro del computador, una unidad de control gestiona los recursos del computador y dirige las prestaciones de sus partes funcionales en respuesta a estas instrucciones.

4. Diagrama de bloques de una computadora



- El CPU **MANDA** direcciones a través del bus de direcciones a los puertos y a la memoria.
- El CPU **MANDA Y RECIBE** datos desde y hacia los puertos y memoria a través del bus de datos.

- **MANDA** señales de control a los puertos y memoria, a través del bus de control.

5. Unidad Central de Procesamiento (CPU)

5.1. Componentes de CPU

- Controla las operaciones del sistema computarizado, **trae el código binario de las instrucciones** desde la memoria, **decodifica** las instrucciones a una serie de acciones simples y **ejecuta** tales acciones.
- El CPU contiene una **unidad aritmética y lógica (ALU)**, la cual realiza operaciones como sumar, restar, or, and, xor, invertir etc., sobre palabras binarias, cuando las instrucciones así lo requieran.
- El CPU también contiene un contador de direcciones o **Contador de Programa** el cual se utiliza para retener la dirección de la próxima instrucción a ser traída desde la memoria.
- Además contiene **registros de propósito general** los cuales se utilizan para almacenar temporalmente datos binarios, y una **circuitería de control** que genera las señales del bus de control.

5.2. Funciones principales del CPU

- Seleccionar, decodificar y ejecutar instrucciones de programa en el orden adecuado.
- Transferir datos hacia y desde la memoria, y hacia y desde las secciones de E/S.
- Responder a interrupciones externas.
- Proporcionar las señales de control y de tiempo necesarias para la totalidad del sistema

6. Jerarquía de Memoria

La jerarquía de memoria se emplea para encontrar un balance entre los usos de memorias, para mantener los costos bajos, sin necesidad de sacrificar el rendimiento del sistema.

Inboard memory	Registers Cache Main memory
Outboard storage	Magnetic disk CD-ROM CD-RW DVD-RW DVD-RAM Blu-Ray
Off-line storage	Magnetic Tape

7. Paginación de memoria (Probablemente no venga)

- Los pedazos de un programa, conocidos como páginas, pueden ser asignados a pedazos de memoria conocidos como frames.
- La memoria principal es dividida en varios pequeños frames de mismo tamaño. Cada proceso es dividido en varias páginas, procesos más pequeños requieren menos páginas, procesos más grandes requieren más páginas.
- El S.O. mantiene una lista de frames libres.
- Así se reduce el espacio de memoria desperdiciado.

8. Arquitectura de la computadora

8.1. Sección de Entrada/Salida

Permite a la computadora tomar datos del mundo real o mandar datos al mundo real. Los periféricos tales como teclados, pantalla e impresoras se conectan a la sección de E/S.

8.2. Bus de direcciones

El bus de direcciones son las lineas de cobre dentro del procesador. Dependiendo de la cantidad de lineas que tenga podremos saber el tamaño de las dirección de memoria máxima direccionable.
El bus de direcciones consiste de 16,20,24 o mas líneas de señales en paralelo. Por estas líneas el CPU envía la localidad de memoria en la cual va escribir o leer.
El número de localidades que el CPU puede direccionar o acceder se determina por el número de líneas del bus de direcciones. Si el CPU tiene N líneas de dirección entonces puede direccionar 2^N localidades.

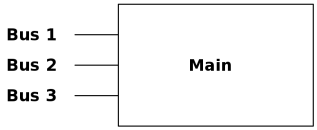


Figura 2: 2^3 direcciones máximas direccionables.

8.3. Bus de datos

El bus de datos consiste de 8,16, 32 o más líneas de señales en paralelo, estas líneas son bidireccionales. Esto significa que el CPU puede leer datos por estas líneas desde la memoria o un puerto así también puede mandar datos a una localidad de memoria o a un puerto.
Muchos dispositivos en un sistema pueden tener sus salidas conectadas al bus de datos, pero solamente uno puede estar habilitado a la vez.

Por lo que cualquier dispositivo conectado al bus de datos debe ser de tres estados de forma que los dispositivos que no estén en uso estén flotados.

Resumen el bus de datos es bidireccional, puede dar y recibir datos el bus de datos es trifásico[1] Leyendo, escribiendo y esperando.

8.4. Bus de Control

- El bus de control consiste de 4 a 10 líneas de señales en paralelo. El CPU manda señales sobre el bus de control para habilitar las salidas de los dispositivos de memoria o puertos direccionados
- Generalmente las señales del bus de control son leer memoria, escribir en memoria, leer E/S y escribir E/S.

9. Diferencias de CISC con RISC

9.1. CISC

- Una computadora con una gran cantidad de instrucciones se clasifica como una Computadora de Conjunto de Instrucciones Complejo, CISC (Complex Instruction Set Computer).
- Durante finales de los años 70 se experimentó con instrucciones muy complejas, posibles gracias al intérprete. Los diseñadores intentaron cerrar la "brecha semántica.[en]tre lo que podían hacer las máquinas y lo que requerían los lenguajes de programación de alto nivel.

9.2. RISC

- Al principio de los 80, muchos diseñadores de computadoras recomendaron que las maquinas utilizaran menos instrucciones con

[1] Que tiene tres estados.

fórmulas más sencillas que pudieran ejecutarse con mayor rapidez dentro del CPU, sin tener que utilizar la memoria con tanta frecuencia.

- Este tipo de computadoras se clasifica como Computadoras de Conjunto de Instrucciones Reducido, RISC (Reduced Instructions Set Computers).
- El concepto de la arquitectura RISC significa un intento para reducir el tiempo de ejecución al simplificar el conjunto de instrucciones de la computadora.

9.3. Tabla de diferencias

RISC

- Instrucciones sencillas en un ciclo
- Cualquier instrucción puede referenciar a memoria
- Procesamiento serie de varias etapas
- Instrucciones ejecutadas por hardware
- Instrucciones de formato fijo
- Pocas instrucciones y modos
- La complejidad está en el compilador
- Varios conjuntos de registros

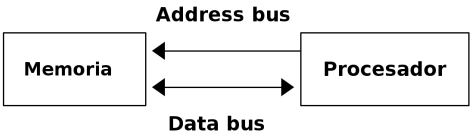
CISC

- Instrucciones complejas en varios ciclos
- Solo LOAD/STORE hacen referencia a memoria
- Poco procesamiento en serie

- Instrucciones interpretadas por un microprograma
- Instrucciones de formato variable
- Muchas instrucciones y modos
- La complejidad está en el microprograma
- Un solo conjunto de registros

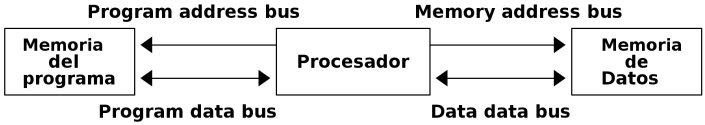
10. Arquitectura de Von Neuman

En la arquitectura Von Neumann el programa y los datos están almacenados en la misma memoria.



11. Arquitectura de Harvard

En la arquitectura Harvard se manejan espacios de memoria separados para almacenar las instrucciones del programa y los datos, por lo que ambas memorias pueden ser accedidas simultáneamente.



12. Taxonomía de Flynn

	Single Instruction	Multiple instructions
Single data	SISD	MISD
Multiple data	SIMD	MIMD

13. Sistema de numeración binario

13.1. Conversión de binario a decimal

El sistema de numeración binario solo consiste de dos símbolos 1 y 0, cada posición corresponde a un exponente numero potencia de 2.

$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	#	DEC
1	0	1	1	1	=	23

13.2. Conversión decimal a binario

Se divide el numero entre 2 y se obtiene su modulo, el resultante del modulo corresponde a cada posición del numero, ejemplo de convertir 200 a binario:

n	200	100	50	25	12	6	3	1
n/2	100	50	25	12	6	3	1	0
n%2	0	0	0	1	0	0	1	1

$$(200)_{10} = (11001000)_2$$

13.3. Suma binaria

```
              1
   35:  0010 0011
+  10:  0000 1010
  ---------------
   45:  0010 1101
```

13.4. Multiplicación binaria

```
25:      0001 1001
 3:      0000 0011
         ---------
         0001 1001  <-  se multiplica 1 por 25
    ...0 0011 001.  <-  otra vez, pero desplazado
    --------------
75:      0100 1011  <- se suman los valores
```

13.5. División entre números binarios

75/5 = 15

```
         1111                  02           0222
    +------------              1001         1000
101 | 1001011                - 101        - 101
       101                     ------       ------
    -------------               100          0011
       1000
        101
    -------------
        0111                    1
         101                    2
    -------------               4
         0101                + 8
          101                  ----
    -------------              15
          000
```

13.6. Números negativos en binario

Representación en signo-magnitud

El bit mas significativo representa el tipo de números, 1 es negativo y 0 es positivo.

Complemento a 1

Niega los valores de los bits, entonces los '0' se vuelven '1' y los '1' se vuelven '0':

```
BIN: 1000101010111  ->  CMP1: 0111010101000
```

Complemento a 2

Es una manera de representar los valores negativos esta representación resuelve el problema del doble cero, pero ahora sus rangos no son iguales en ambas direcciones[2]:

> El rango de números representables en el sistema de complemento a 2 va desde $-(2^{n-1})$ hasta $+(2^{n-1}-1)$, donde 'n' es el numero de bits.

para convertir un numero binario a su complemento a 2, solamente calculamos su complemento a 1 y le sumamos 1:

```
Convertir 5 a CMP2          Convertir -5 a BIN

    5: 0000 0101              -5: 1111 1011 - 1
 CMP1: 1111 1010 + 1        CMP1: 1111 1010
 CMP2: 1111 1011             BIN: 0000 0101 = 5
```

Suma de complemento 2

Si sumamos un numero positivo a un complemento a 2 el equivalente a hacer la resta de los números.

```
         1111 111
 5: .... 0000 0101     Como nuestra operación esta
-5: .... 1111 1011     fija a 8 bits, el resultado
    --------------     es correcto, a pesar de que
    ...1 0000 0000     tiene desbordameniento
```

[2] También utiliza el bit más significativo como bit de signo.

Errores de sobreflujo

> El sobreflujo solo sucede cuando al sumar dos números con el mismo signo, el resultado nos resulta con signo contrario.

```
     No overflow                Overflow

           11   111                   1
   -75:    1011'0101          -92:    1010'0100
+  -25:    1110'0111      +   -40:    1101'1000
  ------------------         ------------------
  -100: 1'1001'1100          -132: 1'0111'1100
```

No overflow A pesar de que el valor abarca in bit extra para la operacion el signo coincide con los valores de la suma, por lo que no hay overflow.

Overflow Por otra parte para solucionar este caso es necesario un bit extra para representalo y el bit de signo es diferente a los operadores, por lo tanto ha y sobreflujo.

13.7. Representación decimal a binario

76.875 ,Para la parte entera de la división hacemos la división como normalmente la hacemos, dividiendo entre 2 y obteniendo el modulo[3].

n	76	38	19	9	4	2	1
n/2	38	19	9	4	2	1	0
n%2	0	0	1	1	0	0	1

Ahora para obtener la parte decimal es necesario que multipliquemos por 2, si el resultado tiene un entero lo eliminamos y lo usamos como el valor binario

[3]En clase no vimos ningún caso con decimal periódico, pero en caso de venir el resultado esta limitado por la cantidad de bits disponible.

n	.375	.75	.50
n*2	0.75	1.5	1
int	0	1	1

$$76{,}375_{10} = 1001100{,}011_2$$

13.8. Binario con punto decimal a flotante

1001100.011, Simplemente extendemos nuestra tabla de exponentes con los números negativos[4]

exp	6	5	4	3	2	1	0		-1	-2	-3
	1	0	0	1	1	0	0	.	0	1	1

$$2^6 + 2^3 + 2^2 + 2^{-1} + 2^{-2} + 2^{-3}$$

$$64 + 8 + 4 + 0{,}5 + 0{,}25 + 0{,}125 = \boxed{76{,}875}$$

13.9. BCD (Binary Coded Decimal)

El BCD sirve para representar los números decimales de manera sencilla en código binario, cada sección del código corresponde a un dígito de nuestro numero:

```
 19 -> 1: 0001 9: 1001
130 -> 1: 0001 3: 0011 0: 0000
```

14. Sistema Hexadecimal

14.1. Ventajas

Este sistema es base 16 por lo que usamos letras para representar los números:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

[4] $x^{-n} = \frac{1}{x^n}$

Las principales ventajas de este sistema son:

- Puede representar números muy grandes en menos espacio.
- Un nibble (4 bits) se pueden representar con un solo valor.

Decimal	Binario	Hexadecimal
217	1101 1001	D9

14.2. Conversión decimal a hexadecimal

Dividimos entre 16 y obtenemos el modulo, el modulo es el valor que corresponde al numero hexadecimal

n	5000	312	19	1
n/16	312	19	1	0
n %16	8	8	3	1
hex(n %16)	8	8	3	1

$$5000_{10} = 1388_{16}$$

14.3. Conversión hexadecimal a decimal

Reemplazamos el valor hexadecimal con su valor decimal, lo multiplicamos por 16^n donde 'n' es la posición:

$$A5A5_{16} = 10 * 16^3 + 5 * 16^2 + 10 * 16 + 5 = 42405_{10}$$

14.4. Conversión binario a hexadecimal

> Un nibble (4 bits) se pueden representar con un solo valor hexadecimal.

Por lo tanto solo tenemos que separar el numero en secciones de 4 bits y sustituir el valor por su correspondiente valor hexadecimal:

```
BIN: 1001 1101 1111 0000
DEC:    9   14   15    0
HEX:    9    E    F    0 -> 0x9EF0
```

Para convertir el binario a hexadecimal basta con hacer el mismo proceso pero a la inversa, convertimos cada dígito en su correspondiente valor binario y lo concatenamos

```
HEX:    9    E    F    0
BIN: 1001 1101 1111 0000 -> 1001110111110000
```

15. Números flotantes

15.1. Distribución de bits de un numero flotante

Precisión simple

	Signo	Exponente	Matiza
Bits	1	8	23

Doble Precisión

	Signo	Exponente	Matiza
Bits	1	11	52

15.2. Convertir decimal a binario

Convertir flotante a decimal

$$200{,}09375$$

n	200	100	50	25	12	6	3	1
n/2	100	50	25	12	6	3	1	0
n %2	0	0	0	1	0	0	1	1

n	0.09375	0.1875	0.75	0.5
n*2	0.1875	0.75	1.5	1
int	0	0	1	1

$$11001000{,}0011_2$$

Desplazamientos

tenemos que recorrer el punto hasta que solo quede un numero al inicio, en nuestro caso el numero de desplazamientos es 7, por lo tanto nuestro exponente es 7.

N. Desp.	Numero
0	11001000.0011
1	1100100.00011
2	110010.000011
3	11001.0000011
4	1100.10000011
5	110.010000011
6	11.0010000011
7	1.10010000011

Calculando el Bias

El exponente esta representado con números negados por lo que tenemos que hacer el ajuste[5], en nuestro caso estamos usando precisión simple por lo que se usan 8 bits:

```
n (n. de bits) = 8
          bias = 2^(8-1) - 1
          bias = 127
```

calculamos el valor inverso de nuestro exponente:

$$N.Desplazamientos + bias = exponente$$

$$7 + 127 = 134_{10} \rightarrow \boxed{10000110_2}$$

[5] $2^{n-1} - 1$, donde 'n' es el numero de bits en el exponente.

Matiza

la matiza es la parte antes del primer uno en nuestra división

$$1.\underbrace{10010000011}_{\text{Matiza}}$$

Representación binaria

signo	Exponente	Matiza
0	1000 0110	1001 0000 0011 0000 0000 000

15.3. Convertir representación binario a decimal

signo	Exponente	Matiza
1	01111000	10001110001110000000000

El primer bit indica el signo del numero, en este caso como el numero es '1' valor es negativo.

Obtener el exponente

```
01111000 = 120

 x + 127 = 120
       x = 120 - 127
       x = -7
```

Convertir la matiza en decimal

Los exponentes son -1, -2, -3... hasta llegar al final:

```
# solucion en python
matiza = "10001110001110000000000"
a = [ 2**(-(i+1)) for i,v in enumerate(matiza) if v=='1' ]
return sum(a)

0.5555419921875
```

Flotante final

$-1 \times 2^{-7} \times (1 + 0{,}5555419921875) = -0{,}01215267181$

16. ¿Que es un traductor?

- Un traductor es un programa que convierte un programa escrito en un lenguaje a otro lenguaje distinto.
- Al lenguaje en que esta escrito el programa original se le denomina lenguaje fuente, y al lenguaje al que se traduce se le conoce como lenguaje objeto.
- Si se contara con un procesador que ejecutara directamente los programas escritos en lenguaje fuente, no sería necesaria la traducción.
- Los programas tanto el fuente como el objetos son equivalentes desde el punto de vista funcional, es decir, producen los mismos resultados.

17. Traducción vs Interpretación

- En la traducción, el programa original en código fuente no es ejecutado directamente. En su lugar, es convertido a un programa equivalente conocido como programa objeto o programa binario ejecutable cuya ejecución comienza solo hasta que la traducción es completada.
- En la interpretación, solo existe un paso: la ejecución del programa fuente original.

18. Lenguajes de alto y bajo nivel

El termino alto nivel significa que están orientados hacia la gente, en contraste con el lenguaje ensamblador de bajo nivel el cual esta orientado hacia la máquina.

19. Estructura de una instruccion de ensamblador

Etiqueta El ensamblador define la etiqueta como equivalente a la dirección en el que se cargará el primer byte del código objeto generado para esa instrucción.

Mnemónico Es el nombre de la operación o función del enunciado.

Operandos Una instrucción en lenguaje ensamblador incluye cero o más operandos. Cada operando identifica un valor inmediato, un valor de registro o una ubicación de memoria.

Comentarios Igual que en otros lenguajes.

19.1. Estructura

```
ETIQUETA:
  MNEMONICO OPERANDO(S) ;COMENTARIO
```

20. Tipos de ensambladores

20.1. de dos pasos

Primera pasada

- El ensamblador solo se ocupa de las definiciones de etiquetas.
- El primer paso se usa para construir una tabla de símbolos que contiene una lista de todas las etiquetas y sus valores de contador de ubicación (LC) asociados.
- El ensamblador debe analizar cada instrucción lo suficiente como para determinar la longitud de la instrucción de máquina correspondiente y, por lo tanto, cuánto incrementar el LC.

Segunda pasada

- Cada instrucción se traduce al código de máquina binario apropiado. La traducción incluye las siguientes operaciones:

20.2. de un paso

Dificultad: Los operandos de instrucción pueden ser símbolos que aún no se han definido en el programa fuente.
Cuando el ensamblador encuentra un operando de instrucción que es un símbolo que aún no está definido, el ensamblador hace lo siguiente:

- 1. Deja el campo del operando de la instrucción vacío (todo ceros) en la instrucción binaria ensamblada.
- 2. El símbolo utilizado como operando se introduce en la tabla de símbolos. La entrada de la tabla está marcada para indicar que el símbolo no está definido.
- 3. La dirección del campo de operando en la instrucción que hace referencia al símbolo indefinido se agrega a una lista de referencias directas asociadas con la entrada de la tabla de símbolos.

21. Linking (Enlazamiento)

21.1. ¿Que es?

El enlazador es un programa que fusiona los espacios de direcciones separados de los módulos de objetos en un único espacio de direcciones.

21.2. ¿Como lo hace?

- 1. Construye una tabla de todos los módulos de objetos y sus longitudes.
- 2. Con base en esta tabla, asigna una dirección de inicio a cada módulo de objeto.

- 3. Encuentra todas las instrucciones que hacen referencia a la memoria y agrega a cada una, una constante de reubicación igual a la dirección inicial de su módulo.
- 4. Encuentra todas las instrucciones que hacen referencia a otros procedimientos e inserta la dirección de estos procedimientos en el lugar

21.3. ¿para que lo hace?

> Como en ensamblador las etiquetas de salto son posisciones de memoria es necesario ajustarlas par que los metodos en el programa tengan sentido.

En cada módulo de objeto, puede haber referencias de direcciones a ubicaciones en otros módulos. Cada una de estas referencias solo puede expresarse simbólicamente en un módulo de objeto no vinculado. El enlazador crea un único módulo de carga que es la unión contigua de todos los módulos de objetos.