75.03 & 95.57 Organización del Computador

U4 – LENGUAJE ENSAMBLADOR
(SEGUNDA PARTE)

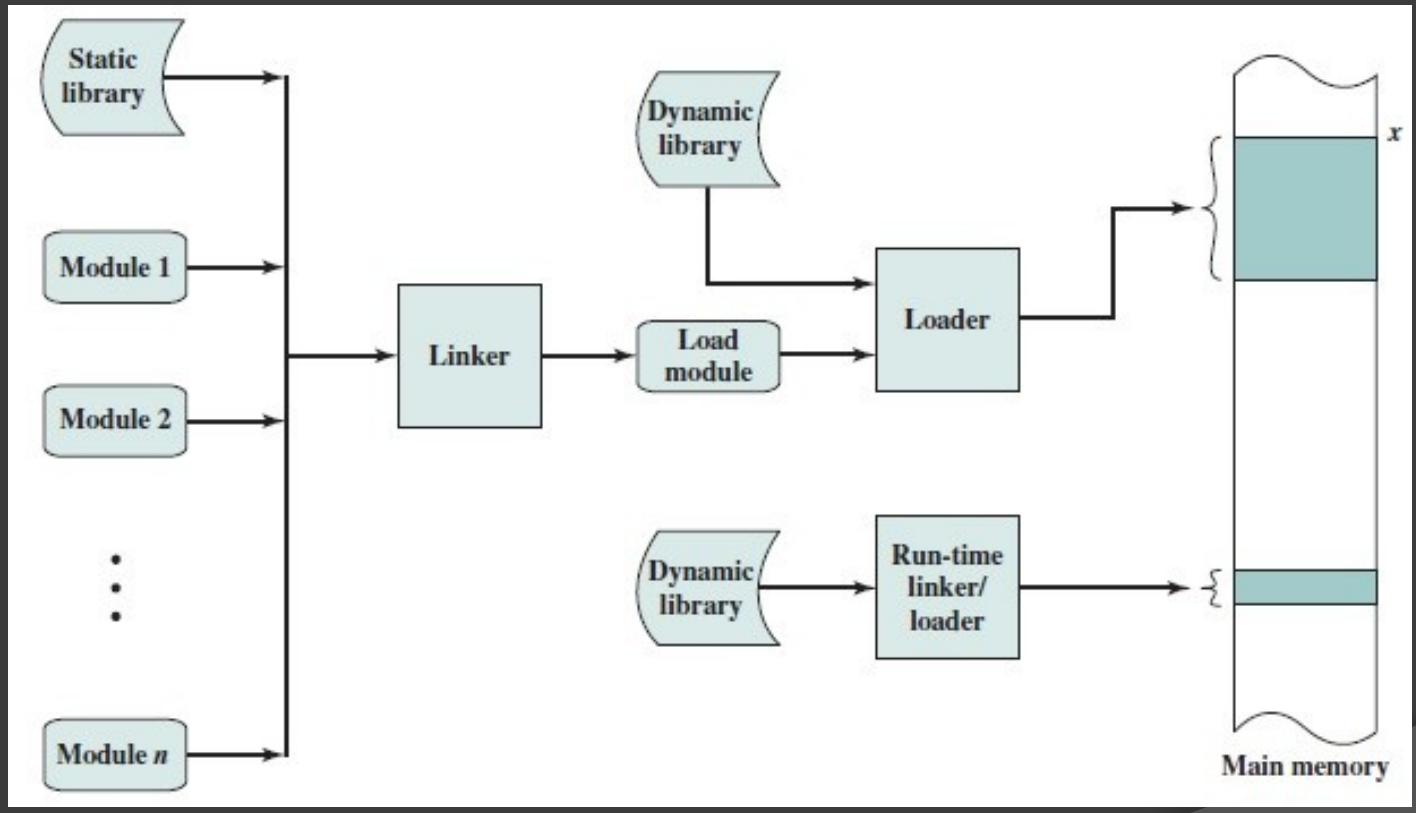
U4 – Lenguaje ensamblador

- Ensamblador
  - Código objeto
    - Definición: “Es la representación en lenguaje de máquina del código fuente programado. Es creado por un compilador o ensamblador y es luego transformado en código ejecutable por el linkeditor”

U4 – Lenguaje ensamblador

- Más allá del ensamblado
  - Linker
    - Definición: “Programa utilitario que combina uno o más archivos con código objeto en un único archivo que contiene código ejecutable o cargable”
  - Loader
    - Definición: “Rutina de programa que copia un ejecutable a memoria principal para ser ejecutado”

U4 – Lenguaje ensamblador



U4 – Lenguaje ensamblador

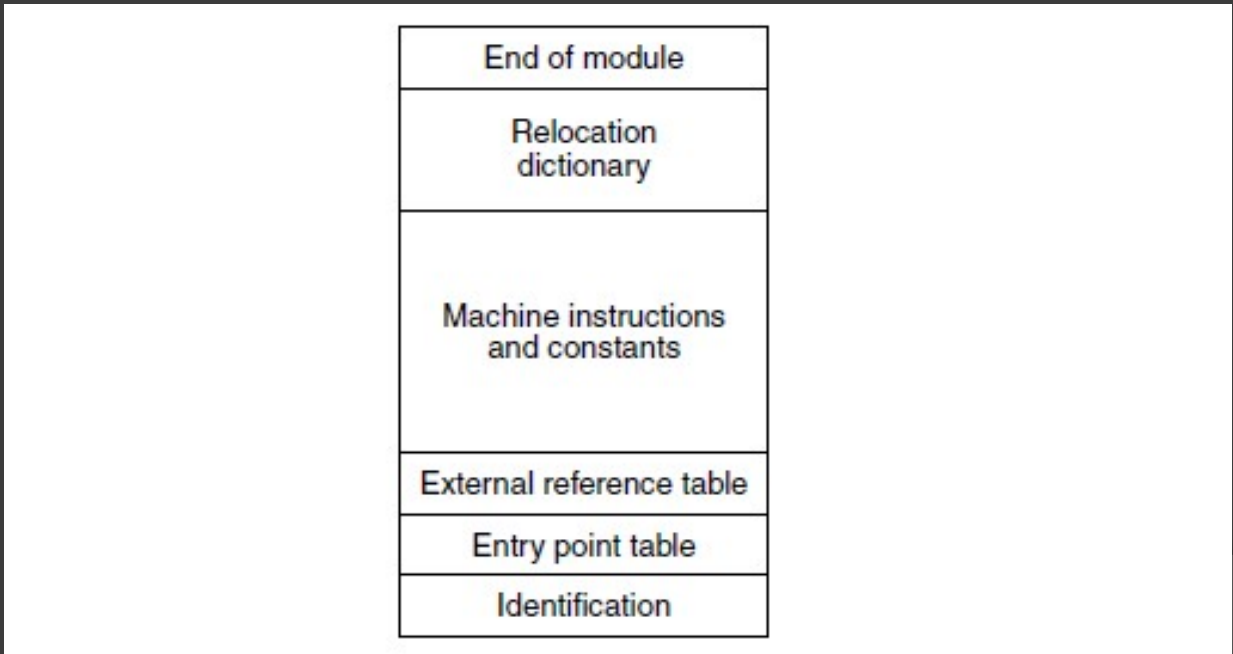
- Dos problemas a resolver
  - Direcciones externas
    - Existen direcciones en el código objeto que no se pueden resolver en tiempo de ensamblado
  - Reubicabilidad
    - ¿Por qué es necesaria?
      - No se sabe que otro programa habrá en memoria a la vez
      - Swap a disco en un entorno de multiprogramación

U4 – Lenguaje ensamblador

- Código objeto
  - Estructura interna
    - Identificación: nombre del módulo, longitudes de las partes del módulo
    - Tabla de punto de entrada: lista de símbolos que pueden ser referenciados desde otros módulos
    - Tabla de referencias externas: lista de símbolos usados en el módulo pero definidos fuera de él y sus referencias en el código
    - Código ensamblado y constantes
    - Diccionario de reubicabilidad: lista de direcciones a ser reubicadas
    - Fin de módulo

U4 – Lenguaje ensamblador

- Código objeto
  - Estructura interna



U4 – Lenguaje ensamblador

- Código objeto
  - Formatos estandarizados
    - OMF (Object Module Format)
    - COFF (Common Object File Format)
    - ELF (Executable and Linkable Format)

U4 – Lenguaje ensamblador

- Linking
  - Estático (linkage editor)
  - Dinámico
    - Load time dynamic linking
    - Run time dynamic linking

U4 – Lenguaje ensamblador

- Linking
  - Estático (linkage editor)
    - Cada módulo objeto compilado o ensamblado es creado con referencias relativas al inicio del módulo
    - Se combinan todos los módulos objeto en un único load module reubicable con todas las referencias relativas al load module

U4 – Lenguaje ensamblador

- Linking estático
  - Módulos objecto

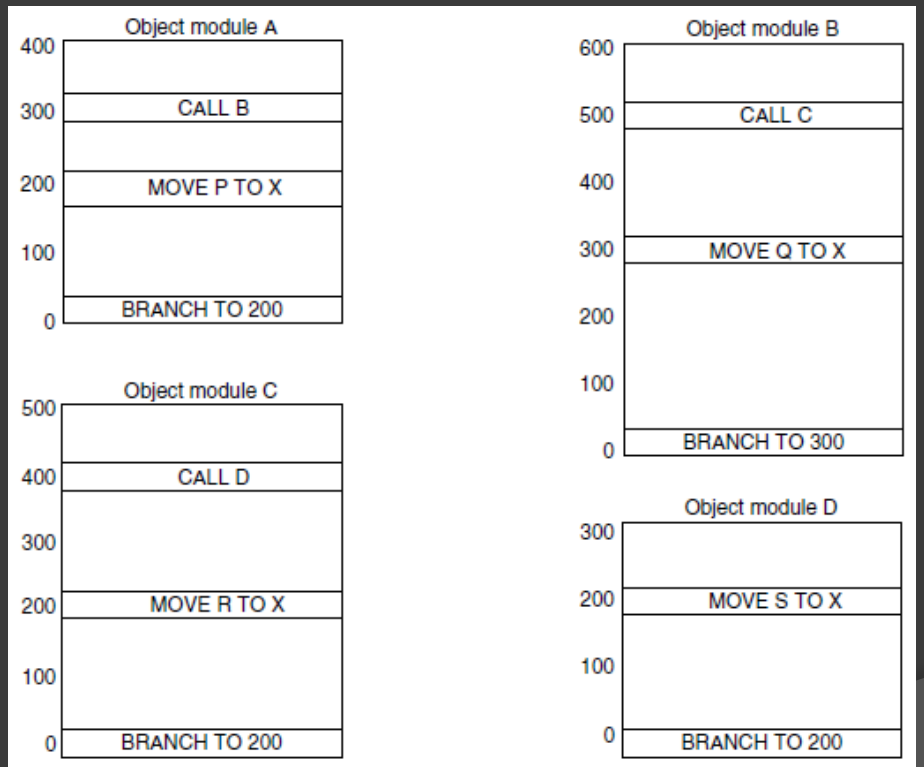


Figure 7-13. Each module has its own address space, starting at 0.

U4 – Lenguaje ensamblador

- Linking estático
  - Generación del load module
    1. Construye tabla de todos los módulos objeto y sus longitudes
    2. Asigna dirección base a cada módulo en base a esa tabla
    3. Busca todas las instrucciones que referencian a memoria y les suma una *constante de reubicación* igual a la dirección de inicio de su módulo objeto
    4. Busca todas las instrucciones que referencian a otros procedimientos e inserta su dirección

U4 – Lenguaje ensamblador

- Linking estático
  - Generación del load module

Module	Length	Starting address
A	400	100
B	600	500
C	500	1100
D	300	1600

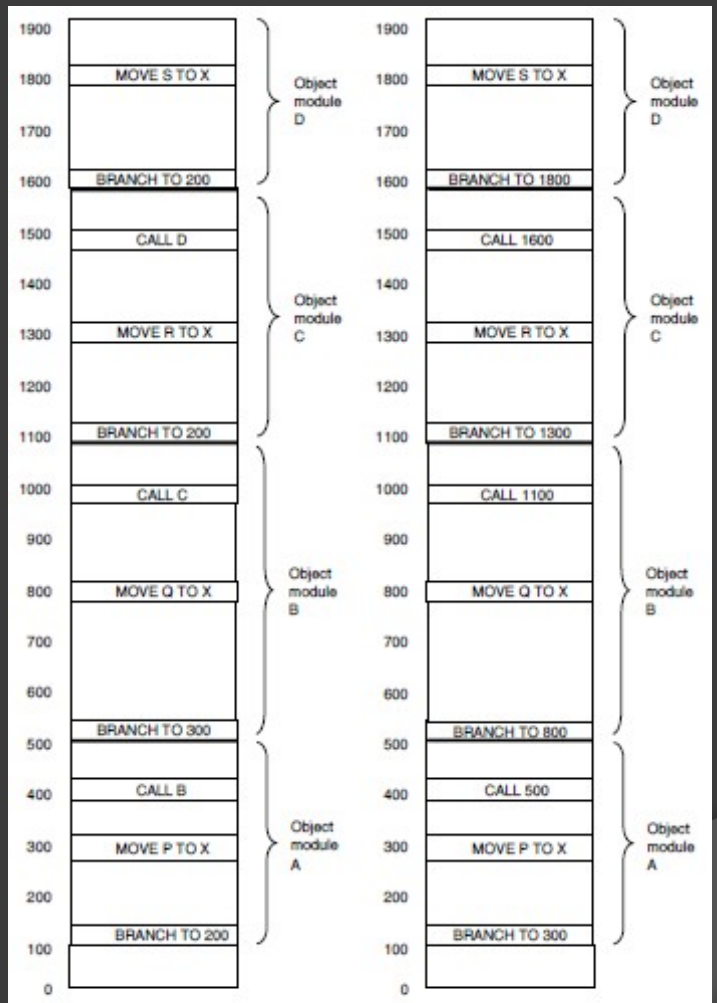


Figure 7-14. (a) The object modules of Fig. 7-13 after being positioned in the binary image but before being relocated and linked. (b) The same object modules after linking and after relocation has been performed.

U4 – Lenguaje ensamblador

- Linking dinámico
  - Se difiere la linkedición de algún módulo hasta luego de la creación del load module
  - Dos tipos:
    - Load time dynamic linking
    - Run time dynamic linking

U4 – Lenguaje ensamblador

- Linking dinámico
  - Load time dynamic linking
    1. Se levanta a memoria el load module
    2. Cualquier referencia a un módulo externo hace que el loader busque ese módulo, lo cargue y cambie la referencia a una dirección relativa desde el inicio del load module
    - Ventajas
      - Facilita la actualización de versión del módulo externo porque no hay que recompilar
      - El sistema operativo puede cargar y compartir una única versión del módulo externo
      - Facilita la creación de módulos de linkeo dinámico a los programadores (ej. Bibliotecas .so en Unix)

U4 – Lenguaje ensamblador

- Linking dinámico
  - Run time dynamic linking
    - Se pospone el linkeo hasta el tiempo de ejecución
    - Se mantienen las referencias a módulos externos en el programa cargado
    - Cuando efectivamente se invoca al módulo externo, el sistema operativo lo busca, lo carga y linkea al módulo llamador.
    - Ventajas
      - No ocupo memoria hasta que la necesito (ej. Bibliotecas DLL de Windows)

U4 – Lenguaje ensamblador

- Loading
  - Loading absoluto
    - El compilador/ensamblador genera direcciones absolutas
    - Solo se puede cargar en un único espacio de memoria
  - Loading reubicable
    - El compilador/ensamblador genera direcciones relativas al LC=0
    - El loader debe sumar un valor X a cada referencia a memoria cuando carga el módulo en memoria
    - El load module tiene que tener información para saber cuales son las referencia a memoria a modificar (diccionario de reubicación)
  - Loading por registro base
    - Arquitecturas que usan registros base para el direccionamiento
    - Se asigna un valor para el registro base asociado a la ubicación en la que se cargó el programa en memoria
  - Loading dinámico en tiempo de ejecución
    - Se difiere el cálculo de las direcciones absolutas hasta que realmente se vaya a ejecutar
    - El load module se carga a memoria con las direcciones relativas
    - La dirección se calcula solo al momento de ejecutar realmente la instrucción (con soporte de hardware especial)

U4 – Lenguaje ensamblador

- Linking (resumen de momentos de linkeo)

(b) Linker

Linkage Time	Function
Programming time	No external program or data references are allowed. The programmer must place into the program the source code for all subprograms that are referenced.
Compile or assembly time	The assembler must fetch the source code of every subroutine that is referenced and assemble them as a unit.
Load module creation	All object modules have been assembled using relative addresses. These modules are linked together and all references are restated relative to the origin of the final load module.
Load time	External references are not resolved until the load module is to be loaded into main memory. At that time, referenced dynamic link modules are appended to the load module, and the entire package is loaded into main or virtual memory.
Run time	External references are not resolved until the external call is executed by the processor. At that time, the process is interrupted and the desired module is linked to the calling program.

U4 – Lenguaje ensamblador

- Loading (resumen de momentos de “binding”)

(a) Loader

Binding Time	Function
Programming time	All actual physical addresses are directly specified by the programmer in the program itself.
Compile or assembly time	The program contains symbolic address references, and these are converted to actual physical addresses by the compiler or assembler.
Load time	The compiler or assembler produces relative addresses. The loader translates these to absolute addresses at the time of program loading.
Run time	The loaded program retains relative addresses. These are converted dynamically to absolute addresses by processor hardware.

U4 – Lenguaje ensamblador

- Referencias
  - “Computer Organization and Architecture – Designing for Perfomance” 9na edición. William Stallings (http://williamstallings.com/ComputerOrganization/)
  - “Structured Computer Organization” 6ta edición. Andrew Tanenbaum / Todd Austin (http://www.pearsonhighered.com/educator/product/Structured-Computer-Organization-6E/9780132916523.page)
  - “Linkers & Loaders” 1ra edición. John R. Levine (https://www.johnlevine.com/index.phtml)