Programación de Sistemas

Unidad 2. Ensamblador de Dos Pasos

Introducción

Definición

- Un ensamblador es un programa que traduce un código fuente en lenguaje ensamblador a un código objeto en lenguaje máquina

- Un ensamblador realiza dos tareas fundamentales:
  - Traducir instrucciones en mnemónico a su equivalente en código objeto
  - Asignar direcciones maquina a las etiquetas utilizadas por el programador

Elementos del Lenguaje Ensamblador

- Cada línea de un programa escrito en lenguaje ensamblador puede contener los siguientes elementos:

  - Etiqueta o símbolo
  - Mnemónico o código de operación
  - Operando (direcciona o etiqueta)
  - Comentarios (especificado por .)

Directivas

- Adicionalmente un código en lenguaje ensamblador puede contener las siguientes directivas:

  - START. Especifica el nombre y dirección de inicio del programa
  - END. Indica el final del programa fuente y (opcionalmente)
  - especifica la primer instrucción ejecutable en el programa
  - BYTE. Genera una constante de un byte de longitud
  - WORD. Genera una constante entera de una palabra de longitud
  - RESB. Reserva el numero indicado de bytes para un dato
  - RESW. Reserva el numero indicado de palabras para un dato

Proceso de Ensamblado

Traducción del Código Fuente

- Para traducir (o ensamblar) un código fuente, es necesario realizar los siguientes pasos:

  - 1 Convertir los códigos mnemónicos a su equivalente en código maquina
  - 2 Convertir los operando simbólicos a su dirección equivalente
  - 3 Construir las instrucciones maquina en un formato adecuado
  - 4 Convertir las constantes especificadas en el programa fuente a su representación interna en la máquina
  - 5 Escribir el programa objeto y posiblemente el listado de ensamblado

Traducción del Código Fuente

- A excepción del segundo paso, el resto puede realizarse procesando el código línea por línea
- Ejemplo:
  - La instrucción:
    - FIRST STL RETADR
  - Al momento de leer la línea, no se conoce la dirección de RETADR

Procesamiento de Instrucciones

- Esta instrucción contiene una referencia adelantada
- La etiqueta (y por lo tanto su dirección) esta definida en una parte posterior del código fuente
- Por esta razón, la mayoría de los ensambladores realizan dos pasos o lecturas sobre el código fuente
- El primer paso asigna direcciones a todas las etiquetas
- El segundo paso realiza la traducción del resto del código fuente

Procesamiento de Directivas

- Además de las instrucciones, se deben procesar otras directivas
- Estas directivas no generan código objeto
- Proporcionan cierta información al ensamblador
- En especial aquellas utilizadas para la declaración de variables o asignación de espacio (BYTE, WORD, RESB, RESW)

Elementos del Código Objeto

Registros Principales

- Encabezado. Contiene el nombre del programa, dirección de inicio y longitud del programa
- Texto. Contiene las instrucciones traducidas y datos del programa
- Final. Indica el fin del programa y la dirección de la primera instrucción ejecutable

Registro de Encabezado

- El registro de Encabezado está formado de la siguiente manera:
  - Columna 1: Símbolo H
  - Columnas 2 a 7: Nombre del programa
  - Columnas 8 a 13: Dirección de inicio (HEX)
  - Columnas 14 a 19: Longitud en bytes (HEX)
- Ejemplo
  - HCOPY 00100000107A

Registro de Texto

- Los registros de Texto están formados de la siguiente manera:
  - Columna 1: Símbolo T
  - Columnas 2 a 7: Dirección del código objeto en ese registro (HEX)
  - Columnas 8 a 9: Longitud del código objeto en ese registro (HEX)
  - Columnas 10 a 69: Código objeto (HEX), dos columnas por byte
- Ejemplo:

T00101E150C10364820610810334C0000454F46000003000000

Registro de Fin

- Los registros de Fin están formados de la siguiente manera:
- Columna 1: Símbolo E
- Columnas 2 a 7: Dirección de la primera instrucción (HEX)
- Ejemplo:

E001000

Estructuras para el Ensamblado

Tablas del Ensamblador

- Un Ensamblador utiliza dos varios tipos de elementos principales para realizar el proceso de ensamblado:

  - Tabla de código de operaciones (OPTAB)
  - Tabla de símbolos (SYMTAB)
  - Contador de localidades (LOCCTR)

Tabla de Código de Operaciones (OPTAB)

- Debe contener (al menos) el mnemónico y su código de operación
- Puede contener información acerca del formato y la longitud de la instrucción
- Una organización conveniente es mediante una tabla hash, utilizando el mnemónico como llave para obtener la posición

OPTAB Durante el Ensamblado

- Durante el Primer paso, las funciones de la OPTAB son:
  - Buscar y validar códigos de operación en el programa fuente
  - Obtener la longitud de la instrucción para actualizar el LOCCTR

- Durante el Segundo paso, las funciones de la OPTAB son:
  - Traducir los mnemónicos a su equivalente en lenguaje máquina

Contador de Localidades (LOCCTR)

- Utilizado como auxiliar en la asignación de direcciones
- Inicializado con la dirección especificada por START
- En caso de que no se especifique, se inicializa con 0 (cero)
- Después de procesar una línea, la longitud de la instrucción o de los datos generados se le suma a LOCCTR
- Cuando se encuentra una etiqueta, LOCCTR proporciona su dirección

Consideraciones para el LOCCTR

- Para el incremento del contador de localidades, es necesario tomar en cuenta las siguientes consideraciones:
  - La longitud de la instrucción (1, 2, 3 o 4 bytes)
  - El efecto de las etiquetas para declarar constantes
  - El efecto de las etiquetas para reservar espacio

Longitud de las Instrucciones

- A partir de la OPTAB se obtiene la longitud de cada una de las instrucciones
- Si se desea utilizar una instrucción de 4 bytes (formato extendido), se debe colocar el símbolo (+) antes de la instrucción
- Ejemplo:
  - FIX (1)
  - DIVR A,X (2)
  - JSUB WRREC (3)
  - +ADD THREE (4)

Ejemplo

- Dadas las siguientes instrucciones, colocar antes de cada una el valor del contador de localidades

COPY	START	1000
FIRST	STL	RETARD
	LDB	#LENGTH
CLOOP	+JSUB	RDREC
	LDA	LENGTH
	COMP	#0
	JEQ	ENDFIL
	+JSUB	WRREC
	J	CLOOP
	END	FIRST

Solución

- El resultado es:

1000	COPY	START	1000
1000	FIRST	STL	RETARD
1003		LDB	#LENGTH
1006	CLOOP	+JSUB	RDREC
100A		LDA	LENGTH
100D		COMP	#0
1010		JEQ	ENDFIL
1013		+JSUB	WRREC
1017		J	CLOOP
		END	FIRST

Efecto de las Directivas

- Las directivas afectan de manera diferente al contador de localidades (LOCCTR)
  - WORD. Agrega 3 bytes al LOCCTR
  - BYTE. Agrega la longitud de la constante al LOCCTR
  - RESW. Agrega $(3 \times \text{operando})_{16}$ al LOCCTR
  - RESB. Agrega $(\text{operando})_{16}$ al LOCCTR

Ejemplo

- Dadas las siguientes instrucciones, colocar antes de cada una el valor del contador de localidades

002D	THREE	WORD	8
	RETADR	RESW	1
	LENGTH	RESW	5
	EOF	BYTE	C'EOF'
	EOF	BYTE	X'F1'
	BUFFER	RESB	4096
	RETADR	WORD	1
		END	FIRST

Solución

- El resultado es:

002D	THREE	WORD	8
0030	RETADR	RESW	1
0033	LENGTH	RESW	5
0042	EOF	BYTE	C’EOF’
0045	EOF	BYTE	X’F1’
0046	BUFFER	RESB	4096
1046	RETADR	WORD	1
		END	FIRST

LOCCTR en el Ensamblado

- Durante el Primer Paso, las funciones del LOCCTR son:
  - Determinar la dirección en la que se encuentran las distintas etiquetas
  - A partir de esta información se genera la tabla de símbolos SYMTAB
- Durante el Segundo Paso, las funciones del LOCCTR son:
  - Auxiliar para el ensamblado del código objeto de las direcciones
  - cuando se utiliza el direccionamiento relativo a contador (*Program Counter*)

Tabla de Símbolos (SYMTAB)

- Incluye el nombre y el valor (dirección) de cada etiqueta en el código fuente
- Contiene banderas para indicar condiciones de error (por ejemplo una etiqueta definida mas de una vez)
- Organizada como una tabla hash, con el símbolo como llave

Generación de SYMTAB

- Es necesario utilizar el contador de localidades LOCCTR
- Su generación se lleva a cabo durante el primer paso del ensamblador
- Debe incluir los valores numéricos de los distintos registros A, B, L, etc
- Es recomendable generar un archivo intermedio que incluya las instrucciones precedidas por el valor del LOCCTR
- Adicionalmente, puede crearse un archivo que contenga la información de la tabla de símbolos

SYMTAB Durante el Ensamblado

- Durante el Primer Paso, la función de la SYMTAB es:
  - Las etiquetas se introducen en la SYMTAB conforme aparecen en el código fuente junto con su dirección

- Durante el Segundo Paso, la función de la SYMTAB es:
  - Los símbolos utilizados como operandos son buscados en SYMTAB para obtener la dirección que será insertada en las instrucciones del ensamblador

Características Dependientes de la Máquina

Introducción

- Características de la arquitectura de la maquina que afectan la codificación de instrucciones
- Existen instrucciones que incluyen el manejo entre registros
- Se debe representar el valor de cada registro en el código objeto

Instrucciones con Registro

- Es necesario representar el numero de cada registro en la instrucción
  - A = 0
  - X = 1
  - L = 2
  - B = 3
  - S = 4
  - T = 5
  - F = 6
  - SW = 9

- Ejemplo:
  - La instrucción:
    - COMPR A,S
  - Se ensambla como:
    - A004

Efecto del Tipo de Direccionamiento

- El programador en la mayora de los casos debe indicar que tipo de direccionamiento se desea utilizar:
  - Direccionamiento inmediato
  - Direccionamiento indirecto
  - Direccionamiento relativo a PC
  - Direccionamiento relativo a base

Consideraciones

- El código de los mnemónicos ocupa 8 bits
- En los formatos 3 y 4 dos de esos bits corresponden a las banderas ***n*** e ***i*** que representan el direccionamiento indirecto e inmediato
- Otros bits, que se localizan en el campo de dirección corresponden a las banderas *x b p e* indican direccionamiento indexado, relativo a base, relativo a PC y modo extendido
- El ajuste realizado a esas banderas afecta la representación de la instrucción, en especial el código de la instrucción

Direccionamiento Indirecto

- Se indica colocando el símbolo @ antes de la dirección/etiqueta
- Las banderas se deben colocar en n=1 i=0

- Ejemplo:
  - Ensamblar la siguiente instrucción en la máquina SIC/XE

ADD @112D

Solución

- El código de la instrucción ADD es 18
- La representación será: 18112D

	ni	xbpe			
0001	1000	0001	0001	0010	1101

- Ajustando el valor de las banderas

	ni	xbpe			
0001	1000	0001	0001	0010	1101

- El ensamblado final es: 1A112D

Direccionamiento Inmediato

- Se indica colocando el símbolo # antes de la dirección/etiqueta
- Las banderas se deben colocar en n=0 i=1
- Ejemplo:
  - Ensamblar la siguiente instrucción en la maquina SIC/XE

ADD # 112D

Solución

- El código de la instrucción ADD es 18
- La representación será: 18112D

	ni	xbpe			
0001	1000	0001	0001	0010	1101

- Ajustando el valor de las banderas

	ni	xbpe			
0001	1001	0001	0001	0010	1101

- De esta manera la instrucción ensamblada es: 19112D

Consideraciones Generales

- En general se puede observar lo siguiente:
  - El direccionamiento inmediato le agrega 1 al código de la instrucción
  - El direccionamiento indirecto le agrega 2 al código de la instrucción

Direccionamiento PC y Base

- La mayora de las instrucciones entre registros y memoria se realizan con direccionamiento relativo a PC o relativo a base
- El programador no especifica cual de los dos se utilizara, sino que es determinado por el mismo ensamblador
- El ensamblador calcula el desplazamiento para ser ensamblado como parte de la instrucción objeto
- De esta manera durante la ejecución del programa se obtiene la dirección correcta
- En base al desplazamiento resultante, se decide el tipo de direccionamiento a utilizar

Tipo de Direccionamiento

- Si el desplazamiento resultante se encuentra entre 0 y 4095, se utilizara el modo relativo a base
- Si el desplazamiento resultante se encuentra entre -2048 y 2047 se utilizara el direccionamiento relativo a PC
- El ensamblador primero verifica si es posible utilizar un tipo de direccionamiento, si no es posible, tratara de utilizar el otro

Direccionamiento Relativo a PC

- El PC (*Program Counter*) siempre contiene la dirección de la SIGUIENTE instrucción
- El desplazamiento para el direccionamiento relativo a PC se calcula:
- Dirección de memoria/etiqueta-dirección del PC
- Ejemplo:
  - Ensamblar la siguiente instrucción:

```
0000    FIRST   STL     RETADR
0003 …
```

Solución

- Considerar que la dirección de la etiqueta RETADR es 0030
- El desplazamiento es: 0030-0003=002D
- Si el código de la instrucción STL es 14, la representación será:

  14002D

- Falta indicar que se tiene direccionamiento relativo a PC en las banderas b=0 y p=1

	ni	xbpe			
0001	0100	0010	0000	0010	1101

- Al realizar el ajuste, el ensamblado es: 14202D

Direcciones Negativas

- Las direcciones negativas se representan utilizando C_2
- Ejemplo:
  - Ensamblar la siguiente instrucción:

    0017 J CLOOP

    001A …

Solución

- Considerar que la dirección de la etiqueta CLOOP es 0006
- El desplazamiento es: 0006-001A=FFEC
- Si el código de la instrucción J es 3C, la representación será: 3CFFEC

	ni	xbpe			
0001	0100	0010	0000	0010	1101

- Falta indicar que se tiene direccionamiento relativo a PC en las banderas b=0 y p=1

	ni	xbpe			
0001	0100	0010	0000	0010	1101

- Al realizar el ajuste, el ensamblado es: 3C2FEC

Direccionamiento Relativo a Base

- El programador indica el valor del registro B
- Carga una etiqueta o valor en el registro B

```
LDB  #LENGTH
```

- Indicar al ensamblador que LENGTH se utilizara como registro base usando la directiva BASE

```
BASE LENGTH
```

- La directiva BASE no genera código objeto
- Ejemplo:
  - Ensamblar la siguiente instrucción: 104E STCH BUFFER

Solución

- Suponer que la dirección de LENGTH es 0033
- Suponer que BUFFER tiene la dirección 0036
- El desplazamiento es: 0036-0033=0003
- Si el código de la instrucción STCH es 54, la representación sería: 540003

	ni	xbpe			
0101	0100	0000	0000	0000	0011

- Falta indicar que se tiene direccionamiento relativo a base en las banderas b=1 y p=0 quedando: 544003

	ni	xbpe			
0101	0100	0100	0000	0000	0011

Consideraciones

- El direccionamiento relativo a PC coloca en 2 el dígito hexadecimal mas significativo de la dirección
- El direccionamiento relativo a Base coloca en 4 el dígito hexadecimal mas significativo de la dirección
- Es necesario ajustarlas banderas para indicar el direccionamiento relativo a PC cuando se utiliza direccionamiento indirecto

Modo Extendido

- Se especifica utilizando el símbolo + antes de la instrucción
- Esto indica que se utilizaran 4 bytes para el código objeto
- La dirección especificada o la asignada a una etiqueta no sufre modificaciones
- Ejemplo:
  - Ensamblar la instrucción: CLOOP +JSUB RDREC

Solución

- Suponer que la dirección de RDREC es 1036
- El código de la instrucción JSUB es 48
- Se colocan 0 para completar los 20 bits de la dirección
- De esta manera la representación será: 48001036

	ni	xbpe					
0100	1000	0000	0000	0001	0000	0011	0110

- Falta indicar que se utiliza el formato extendido en la bandera e=1, esto coloca en 1 el dígito hexadecimal mas significativo de la dirección quedando: 48101036

	ni	xbpe					
0100	1000	0001	0000	0001	0000	0011	0110

Combinando Direccionamientos

- Es posible tener distintas combinaciones de los modos de direccionamiento y formatos
- Ejemplo:
  - Ensamblar la siguiente instrucción: +LDT #4096

Solución

- La dirección es $(4096)_{10}$
- El código de la instrucción LDT es 74
- $(4096)_{10} = (1000)_{16}$
- De esta manera la representación será: 74001000

	ni	xbpe					
0111	0100	0000	0000	0001	0000	0000	0000

- Falta indicar que se utiliza el formato extendido y el direccionamiento inmediato en la banderas e=1 n=0 i=1 quedando: 75101000

	ni	xbpe					
0111	0101	0001	0000	0001	0000	0000	0000

Elementos sin Código Objeto

- Las siguientes directivas NO producen código objeto:
  - START
  - BASE
  - RESW
  - RESB
  - END

Relocalización de Programas

Introducción

- En ocasiones es deseable tener ejecutándose mas de un programa a la vez
- Si se conoce que programas se estarán ejecutando, es posible asignar direcciones para que sean ensamblados para que no se traslapen
- No se sabe exactamente que trabajos serán procesados o su longitud
- La dirección inicial del programa no se conoce hasta el momento en que se carga

Programa Absoluto

- Un programa al que se le indica en que dirección comenzar se conoce como programa absoluto (o ensamblado absoluto)
- Ejemplo:

```
COPY    START   1000
```

- Este programa debe ser cargado en la dirección 1000, en caso contrario no funcionara de manera adecuada

Ejemplo

- Considerar la siguiente línea de código:

  101B LDA THREE 00102D

- El registro A será cargado de la localidad de memoria 102D (localidad de THREE)
- Si se desea ejecutar el programa empezando en la dirección
- 2000, la dirección 102D no contendrá el valor deseado
- Se podría invadir espacio ocupado por otro programa

Programas Relocalizables

- Es necesario realizar algunos cambios en el campo de dirección para que el programa pueda ser cargado y ejecutado a partir de cualquier dirección
- Hay partes del programa que no dependen de la dirección en la que sea cargado, por ejemplo la generación de constantes
- Se pueden identificar las partes de un programa objeto que necesitan ser modificadas
- Un programa objeto que contiene la información necesaria para realizar este tipo de modificaciones es llamado un programa relocalizable

Relocalización

- El programa inicia en la dirección 0000
- La instrucción +JSUB RDREC es cargada en la dirección 0006
- La dirección de la etiqueta RDREC es 1036
- Se desea que no importando en donde se inicie el programa, RDREC siempre este 1036 bytes después de la dirección de inicio

Relocalización

- Cuando el ensamblador genera el código objeto para una instrucción, insertara la dirección relativa al inicio del programa de la etiqueta
- Producirá un comando para el cargador, indicándole que se debe sumar la dirección de inicio del programa al campo de dirección en la instrucción al momento de realizar la carga
- Es necesario el uso de un nuevo registro, el registro de modificación M

Registro de Modificación

- La estructura de los Registros de Modificación son:
  - Columna 1: M
  - Columnas 2 a 7: Dirección inicial del campo a ser modificado relativo al inicio del programa (hexadecimal)
  - Columnas 8 a 9: Longitud del campo de dirección a ser modificado, en mitad de bytes (hexadecimal)
- La longitud es almacenada en mitad de bytes debido a que los campos de dirección que serán modificados no ocupan un numero entero de bytes

Generación del Registro M

- Ejemplo:
  - El programa comienza en la dirección 0000
  - Considerar la instrucción: 0006 +JSUB RDREC
  - Si la dirección de RDREC es 1036 (TABSYM), el ensamblado es: 48101036
  - Determinar el valor del registro de modificación M

Solución (I)

- Almacenamiento en memoria
- El código 48101036, en memoria esta almacenado de la siguiente manera, debido a que cada dirección de memoria puede contener solo 1 byte (8 bits)

0006	48
0007	10
0008	10
0009	36

- A partir de esta representación, se obtiene el registro M00000705

Solución (II)

- Los dos primeros dígitos (8 bits) que se encuentran en la dirección 0006 corresponden al código de operación, por esa razón la modificación se indica a partir de la dirección 0007
- Esto en el registro de modificación esta representado con (000007)

M00000705

Solución (III)

- El primer dígito después del código de operación corresponde a las banderas, en este caso e=1
- Los tres primeros dígitos (481) no requieren modificación
- Esto representa 12 bits, de los 32 disponibles, quedando solo 20, los cuales representan 5 medios bytes (medio byte = 4bits, por 5 = 20 bits), por lo tanto la longitud del campo se indica como 05 medios bytes (05)

- M00000705

Desventajas de los Registros de Modificación

- Suponer que se trabaja en una arquitectura que no admite direccionamiento, considerar el siguiente bloque de código.

0000	COPY	START	0
0000	FIRST	STL	RETADR
0003	CLOOP	JSUB	RDREC
0006		LDA	LENGTH
0009		RSUB	4C000
000C		JEQ	ENDFIL
000F	EOF	BYTE	C'EOF'
0012	THREE	WORD	3
0015	RETADR	RESW	1
0018	ENDFIL	LDA	EOF

Desventaja de Registros M

- En el código anterior, la mayoría de las instrucciones requieren relocalizarse:

0000	COPY	START	0
0000	**FIRST**	**STL**	**RETADR**
0003	**CLOOP**	**JSUB**	**RDREC**
0006		**LDA**	**LENGTH**
0009		RSUB	4C000
000C		**JEQ**	**ENDFIL**
000F	EOF	BYTE	C'EOF'
0012	THREE	WORD	3
0015	RETADR	RESW	1
0018	**ENDFIL**	**LDA**	**EOF**

Desventaja de Registros M

- Cada una de estas instrucciones generaría un registro de Modificación que dependa de la dirección de inicio por ejemplo:
  - M000000
  - M000003
  - M000006
  - M00000C
  - M000018

Máscaras de Bits

- En este método no existen los registros de Modificación sino que se agrega información a los registros de texto
- Esta información es conocida como bits de relocalización y está asociada a cada palabra en el código objeto
- En el caso en donde cada instrucción ocupa una palabra, existe un bit para cada posible instrucción
- Los bits de relocalización usualmente son agrupados en una máscara de bits representada en hexadecimal que se coloca después del indicador de longitud en cada registro de Texto

Ejemplo

0000	COPY	START	0	
0000	FIRST	STL	RETADR	140033
0003	CLOOP	JSUB	RDREC	481030
0006		LDA	LENGTH	000036
0009		COMP	ZERO	280030
000C		JEQ	ENDFIL	300015
000F		JSUB	WRREC	481061
0012		J	CLOOP	3C0003
0015	ENDFIL	LDA	EOF	00002A
0018		STA	BUFFER	0C0039
0019		LDA	THREE	00002D

Ejemplo

- Se generará el siguiente código objeto

HCOPY 00000000107A

T0000001E**FFC**1400334810300000362800303000154810613C
000300002A0C003900002D

- En dónde **FFC** representa los valores de la máscara de bits

Generación de la Máscara de Bits

- Como el código contiene 10 palabras, es decir 10 instrucciones con longitud de una palabra, se requieren indicar que esas diez instrucciones necesitan relocalización

- Esto se logra colocando 10 unos (uno por cada palabra): **1111111111**

- Como se requieren dígitos hexadecimales, hay que completar los 12 bits, de esta manera se obtiene **111111111100** que en hexadecimal representan **FFC**

Consideraciones

- Como cada bit de relocalización está asociado a un segmento de 3 bytes de código, cualquier valor que vaya a ser modificado durante la relocalización debe coincidir con alguno de estos bytes

- Si esto no sucede, será necesaria la creación de un nuevo registro de Texto, incluso si aún queda espacio suficiente en el que se está trabajando

Ejemplo

- Considerar la siguiente fracción de código fuente

1057	EXIT	STX	LENGTH	100036
105A		RSUB		4C0000
105D	INPUT	BYTE	X'F1'	F1
105E	MAXLEN	WORD	4096	001000
1061	WRREC	LDX	ZERO	040030

Generación de Máscara de Bits

- Debido a que la instrucción ensamblada como F1 no es de longitud de una palabra, la asignación en las máscaras de un bit por palabra se ve afectada

- Como la siguiente instrucción 001000 no requiere relocalización, puede considerarse, en caso contrario ya no se podría incluir

- Cuando aparece otra instrucción ensamblada que si requiere relocalización, se debe crear un nuevo registro de texto a pesar de que todavía había espacio en el actual

Relocalización No Necesaria

- Existen casos en donde la relocalización no es necesaria para el correcto funcionamiento del programa:
  - Direccionamiento relativo a PC
  - Direccionamiento relativo a base
  - Direccionamiento indirecto

Bloques de Programa

Introducción

- Hasta el momento los programas han sido ensamblados como una sola unidad
- La mayora de los ensambladores permiten un manejo mas flexible del código fuente
- Esto permite que las instrucciones maquina generadas aparezcan en un orden diferente del que fueron escritas en el código objeto
- Esto lleva a la creación de numerosas partes independientes del código objeto
- Estas partes mantienen su identidad y son manejadas por separado por el cargador

Definición de Bloque de Programa

- Bloque de programa. Segmentos de código reordenados dentro de una sola unidad de programa objeto
- Secciones de control. Segmentos que son traducidos a unidades de programa independientes

Características de Bloque de Programa

- Cada bloque de programa puede contener diferentes segmentos de código
- El ensamblador deberá reordenar estos segmentos para reunir las piezas de un mismo bloque
- Estos bloques serán asignados a sus direcciones correspondientes en el código objeto
- Los bloques aparecerán en el mismo orden en el que aparecen en el código fuente

Definiendo Bloques de Programa

- Al inicio del programa todas las instrucciones pertenecen a un bloque sin nombre
- La directiva USE indica que porción del código fuente corresponde a cada bloque, al comienzo del programa
- La directiva USE sin un nombre como parámetro también hace referencia a este bloque

Código con Bloques de Programa

```
0000    0       COPY    START   0
0000    0       FIRST   STL     RETADR
0003    0       CLOOP   JSUB    RDREC
0006    0               LDA     LENGTH
0009    0               COMP    #0
000C    0               JEQ     ENDFIL
000F    0               JSUB    WRREC
0012    0               J       CLOOP
0015    0       ENDFIL  LDA     =C'EOF'
0018    0               STA     BUFFER
001B    0               LDA     #3
001E    0               STA     LENGTH
0021    0               JSUB    WRREC
0024    0               J       @RETADR
0000    1               USE     CDATA
0000    1       RETADR  RESW    1
0003    1       LENGTH  RESW    1
0000    2               USE     CBLKS
0000    2       BUFFER  RESB    4096
1000    2       BUFEND  EQU     *
1000    2       MAXLEN  EQU     BUFEND-BUFFER
                        USE
0027    0               LDA     WRREC
002A    0               COMP    #5
002D    0               JEQ     BUFFER
0030    0               J       ENDFIL
0033    0               JSUB    CLOOP
(0036)                  END     FIRST
```

Proceso de Ensamblado

- Efecto de Bloques de Programas en el Contador de Localidades
  - Durante el paso 1 existe un contador de localidades para cada bloque
  - El contador de localidades para un bloque es inicializado a 0 cuando el bloque empieza
  - El valor actual del contador para ese bloque es guardado cuando ocurre un cambio de bloque y se carga cuando se encuentra de nuevo el mismo bloque

Primer Paso del Ensamblador

- A cada etiqueta en el código fuente se le asigna una dirección que es relativa al comienzo del bloque que la contiene
- Las etiquetas son introducidas a la tabla de símbolos y el numero o nombre del bloque al que pertenecen es almacenado junto con la dirección relativa que les fue asignada
- Al finalizar el paso 1, los últimos valores del contador de localidades para cada bloque indican la longitud de ese bloque
- Entonces el ensamblador puede asignar a cada bloque una dirección inicial en el código objeto

Tabla de Bloques

Nombre Bloque	Número Bloque	Dirección	Longitud
(default)	0	0000	0036
CDATA	1	0036	0006
CBLKS	2	003C	1000

Segundo Paso del Ensamblador

- El ensamblador necesita la dirección para cada símbolo relativo al inicio del código objeto (no a cada bloque individual)
- Esta dirección se encuentra en la tabla de símbolos
- El ensamblador añade la dirección del símbolo, relativo al inicio del bloque, a la dirección del bloque en donde se encuentra

Ejemplo

- Ensamblar la siguiente dirección
  - 0006 0 LDA LENGTH
  - En la tabla de símbolos se encuentra que la dirección de LENGTH es 0003 relativa al inicio del bloque CDATA
  - La dirección de inicio de CDATA es 0066
  - La dirección destino es 0066 + 0003 = 0069
- El ensamblador no reordena el código objeto. Solo asegura que cada registro de texto contiene la dirección de inicio adecuada en base a su posición según la Tabla de Bloques

Secciones de Control

- Parte de un programa que mantiene su identidad después de ser ensamblada
- Cada sección de control puede ser cargada y relocalizada independientemente de otras secciones
- Es posible tener instrucciones en una sección que hagan referencia a instrucciones o datos localizados en otras secciones
- Estas referencias entre secciones de control son llamadas referencias externas

Declaración

- Para declarar las secciones de control se utiliza la directiva CSET

NOMBRE CSET

- Esto indica que todas las instrucciones, operandos y símbolos a partir de esa sentencia pertenecen a la sección llamada NOMBRE

Secciones de Control y Bloques de Programa

- La diferencia entre una sección de control y un bloque de programa es que cada sección es manejada de manera independiente por el ensamblador, en ocasiones incluso no es necesario que todas las secciones de control sean ensambladas al mismo tiempo

Definición de Símbolos

- Los símbolos que son definidos en una sección de control no pueden ser utilizados directamente por otra
- Definición Externa
  - Se realiza con la directiva EXTDEF
  - Se utiliza para declarar símbolos, llamados símbolos externos que son declarados en una sección y serán utilizados por otras
  - Los nombres de las secciones de control no necesitan ser declarados con EXTDEF ya que son considerados automáticamente como símbolos externos
- Referencia Externa
  - Se realiza con la directiva EXTREF
  - La declaración EXTREF declara símbolos que son utilizados en una sección de control, pero que fueron definidos en otra

Símbolos Repetidos

- Utilizando bloques de programa es posible tener símbolos definidos mas de una vez en el mismo programa
- Es posible definir el símbolos con el mismo nombre siempre y cuando se declaren en secciones distintas
- Al momento de crear la tabla de símbolos, se debe añadir como información la sección en la que se encuentra el símbolo

Generación de Código Objeto

- El ensamblador debe incluir información en el programa objeto para que el cargador inserte los valores adecuados cuando sea necesario
- Es necesario el uso de dos nuevos tipos de registros: el registro DEFINE (Definición) y REFER (Referencia)
- Cuando se genera el código objeto, se obtiene un código para cada sección como si fueran programas independientes, con registros de encabezado y registros de n para cada sección

El Registro DEFINE

- Proporciona información acerca de símbolos externos que se definen en una sección de control, es decir, aquellos declarados con EXTDEF
- Composición
  - Columna 1:D
  - Columnas 2 a 7:Nombre del símbolo externo definido en la sección
  - Columnas 8 a 13:Direccion relativa del símbolo dentro de la sección (hex)
  - Columnas 14 a 73:Informacion de las columnas 2 a 13 para otros símbolos externos

El Registro REFER

- Muestra símbolos que son usados como referencias externas, aquellos definidos con EXTREF
- Composición:
  - Columna 1:R
  - Columnas 2 a 7:Nombre del símbolo externo referenciado en la sección
  - Columnas 8 a 73:Nombre de otros símbolos externos a los que se hace referencia

El Registro de Modificación

- Otro registro que necesita nueva información es el registro de Modificación
- Composición:
  - Columna 1:M
  - Columnas 2 a 7:Direccion de inicio del campo a ser modificado, relativo al comienzo de la sección
  - Columnas 8 a 9:Longitud del campo a ser modificado, en medios bytes
  - Columna 10:Bandera de modificación (+ o -)
  - Columnas 11 a 16:Smbolos externos cuyo valor será sumado o restado del campo indicado

Ejemplo

- Considerar la instrucción

CLOOP +JSUB WRREC

- Que se encuentra en la Sección COPY y utiliza a WRREC como símbolo externo declarado con EXTREF
- Ensamblando se tiene:

48100000

- En el campo de operando se coloca 0000 ya que no se conoce la dirección de WRREC
- Se genera el registro de Modificación

M00000405 + WRREC