



Tecnológico Nacional de México Campus Querétaro

LENGUAJE DE INTERFAZ

Tarea Macros, unidad 3

Que presenta:

Perez Nuñez Jared Giezi

Romero Dávila José María

Estudiantes de la carrera:

Ingeniería en sistemas computacionales

Profesor:

RUBEN ARIAS AGUILAR

Numero de lista:

**22140230
22140248**

Grupo

6^a

Descripción

Investigar que son los Macros y hacer 4 ejercicios de: Internas, Externas, Proc y MACORS INT, PROC INT Y MAC EXT, PROC EXT
CONCLUSIONES

Macros en Emu8086

- Una macroinstrucción no es más que una conveniencia notacional para el programador (versión abreviada).
- Cada vez que se llama una macro en un programa, el ensamblador, en lugar de la llamada a la macro, copia y pega el código real de la misma (MACROEXPANSIÓN).
- Las funciones de un procesador de macros implican la sustitución de un grupo de caracteres o líneas por otras.
- El diseño de un procesador de macros no está directamente relacionado con la estructura del computador en el que se va a ejecutar.
- El uso más común de los procesadores de macros es en la programación en lenguaje ensamblador,
- también se pueden utilizar procesadores de macros con lenguajes de programación de alto nivel,
- hay procesadores de macros de aplicación general que no están ligados a ningún lenguaje en particular.

Funciones básicas del procesador de macros

- Definición,
- invocación y
- expansión de macros
- Ejemplo: programa en ensamblador con macros.
- Este programa define y usa dos instrucciones a macros: RDBUFF y WRBUFF.

```

5 COPY START 0 COPIA EL ARCHIVO DE LA ENTRADA A LA SALIDA
10 RBUFF MACRO &INDEV,&BUFADR,&RECLTH
15 -
20 - MACRO QUE LEE UN REGISTRO EN EL BUFFER
25 -
30 CLEAR X LIMPIA EL CONTADOR DE CICLO
35 CLEAR A
40 CLEAR S
45 +LDT #4096 ASIGNA LA LONGITUD MAXIMA DEL REGISTRO
50 TD =>X'&INDEV' PRUEBA EL DISPOSITIVO DE ENTRADA
55 JEQ *-3 REPITE EL CICLO HASTA QUE ESTE LISTO
60 RD =>X'&INDEV' LEE EL CARÁCTER EN EL REGISTRO A
65 COMPR A,S EXAMINA SI HAY FIN DE REGISTRO
70 JEQ *-11 SALE DEL CICLO SI ES FIN DE REGISTRO
75 STCH $BUFADR,X ALMACENA EL CARÁCTER EN EL BUFFER
80 TTX T REPITE EL CICLO A MENOS QUE SE HAYA
85 JLT *-19 ALCANZADO LA LONGITUD DEL REGISTRO
90 STX &RECLTH GUARDA LA LONGITUD DEL REGISTRO
95 MEND
100 WRBUFF MACRO &OUTDEV,&BUFADR,&RECLTH
105 -
110 - MACRO QUE ESCRIBE EL REGISTRO DEL BUFFER
115 -
120 CLEAR X LIMPIA EL CONTADOR DE CICLO
125 LDT &RECLTH
130 LDCH &BUFADR,X TOMA EL CARACTER DEL BUFFER
135 TD =>X'&OUTDEV' PRUEBA DEL DISPOSITIVO DE SALIDA
140 JEQ *-3 REPITE EL CICLO HASTA QUE ESTE LISTO
145 WD =>X'&OUTDEV' ESCRIBE EL CARACTER
150 TTX T REPITE EL CICLO HASTA QUE SE HAYAN
155 JLT *-14 ESCRITO TODOS LOS CARACTERES
160 MEND
165 -
170 - PROGRAMA PRINCIPAL
175 -
180 FIRST STL RETADR GUARDA LA DIRECCIÓN DE RETORNO
190 CLOPP RBUFF F1,BUFFER,LENGTH LEE EL REGISTRO DE ENTRADA EN EL BUFFER
195 LDA LENGTH VERIFICA SI ES FIN DE ARCHIVO
200 COMP #0
205 JEQ ENDFIL SALE SI ENCONTRO EL FIN DE ARCHIVO
210 WRBUFF 05,BUFFER,LENGTH ESCRIBE EL REGISTRO EN LA SALIDA
215 J CLOOP CICLO
220 ENDFIL WRBUFF 05,EOF,THREE INSERTA MARCA DE FIN DE ARCHIVO
225 J @RETADR
230 EOF BYTE C'EOF'
235 THREE WORD 3
240 RETADR RESW 1
245 LENGTH RESW 1 LONGITUD DEL REGISTRO
250 BUFFER RESB 4096 AREA DE BUFFER DE 4096 BYTES
255 END FIRST

```



5	COPY	START	0	COPIA EL ARCHIVO DE LA ENTRADA EN LA SALIDA
180	FIRST	STL	RETADR	GUARDA LA DIRECCION DE RETORNO
190	.CLOOP	RDBUFF		F1,BUFFER,LENGTH LEE EL REGISTRO DE ENTRADA EN EL BUFFER
190a	CLOOP	CLEAR	X	LIMPIA EL CONTADOR DE CICLO
190b		CLEAR	A	
190c		CLEAR	S	
190d		+ LDT	#4096	ASIGNA LA LONGITUD MAXIMA DEL REGISTRO
190d		TD	=X'F1'	PRUEBA EL DISPOSITIVO DE ENTRADA
190e		JEQ	*-3	REPITE EL CICLO HASTA QUE ESTE LISTO
190f		RD	=X' F1'	LEE EL CARACTER EN EL REGISTRO A
190g		COMPR	A,S	EXAMINA SI HAY FIN DE REGISTRO
190h		JEQ	*+11	SALE DEL CICLO SI ES FIN DE REGISTRO
190i		STCH	BUFFER,X	ALMACENA EL CARACTER EN EL BUFFER
190j		TIXR	T	REPITE EL CICLO A MENOS QUE SE HAYA
190k		JLT	*-19	ALCANZADO LA LONGITUD MAXIMA
190l		STX	LENGTH	GUARDA LA LONGITUD DEL REGISTRO
195		LDA	LENGTH	VERIFICA SI ES FIN DE ARCHIVO
200		COMP	#0	
205		JEQ	ENDFIL	SALE SI ENCONTRO EL FIN DE ARCHIVO
210		WRBUFF		05,BUFFER,LENGTH ESCRIBE EL REGISTRO EN LA SALIDA
210a		CLEAR	X	LIMPIA EL CONTADOR DE CICLO
210b		LDT	LENGTH	
210c		LDCH	BUFFER,X	TOMA EL CARACTER DEL BUFFER
210d		TD	=X'05'	PRUEBA EL DISPOSITIVO DE SALIDA
210e		JEQ	*-3	REPITE EL CICLO HASTA QUE ESTE LISTO
210f		WD	=X'05'	ESCRIBE EL CARÁCTER
210g		TIXR	T	REPITE EL CICLO HASTA QUE SE HAYAN
210h		JLT	*-14	ESCRITO TODOS LOS CARACTERES
215		J	CLOOP	CICLO
220	.ENDFIL	WRBUFF	05,EOF,THREE	INSERTA MARCA DE FIN DE ARCHIVO
220a	ENDFIL	CLEAR	X	LIMPIA EL CONTADOR DE CICLO
220b		LDT	THREE	
220c		LDCH	EOF,X	TOMA EL CARACTER DEL BUFFER
220d		TD	=X'05'	PRUEBA EL DISPOSITIVO DE SALIDA
220e		JEQ	*-3	REPITE EL CICLO HASTA QUE ESTE LISTO
220f		WD	=X'05'	ESCRIBE EL CARÁCTER
220g		TIXR	T	REPITE EL CICLO HASTA QUE SE HAYAN
220h		JLT	*-14	ESCRITO TODOS LOS CARACTERES
225		J	@RETADR	
230	EOF	BYTE	C'EOF'	
235	THREE	WORD	3	
240	RETADR	RESW	1	
245	LENGTH	RESW	1	LONGITUD DEL REGISTRO
250	BUFFER	RESB	4096	AREA DE BUFFER DE 4096 BYTES
255		END	FIRST	

Una proposición de invocación a macros se suele denominar una macrollamada. Para evitar confusiones con la proposición de llamada utilizada en los procedimientos y subrutinas, es preferible usar el término

- Los argumentos y los parámetros se asocian entre sí, de acuerdo con sus posiciones.
 - Las dos invocaciones de WRBUFF especifican argumentos distintos, por lo que producen expansiones distintas.
 - Las proposiciones de invocación a macros se tratarán como comentarios, y las instrucciones generadas de las expansiones de macros se ensamblarán igual que si el programador las hubiera escrito directamente.
 - Las macroinstrucciones se han escrito para que el cuerpo de la macro no contenga etiquetas.
 - Para evitar la duplicación de símbolos se han eliminado las etiquetas del cuerpo de definición de estas macros.
 - "JLT *-14" ◇práctica de programación pobre. Tablas y lógica del procesador de macros
- Procesador de macros de dos pasos: 1. Procesar las definiciones de macros 2. Las proposiciones de invocación a macros se expanden durante el segundo paso.

- No permite que el cuerpo de una macroinstrucción contenga definiciones de otras macros


1	MACROS	MACRO	{Define las macros para la versión estándar de SIC}	
2	RDBUFF	MACRO	&INDEV,&BUFADR,&RECLTH	
		.	{Versión estándar de SIC}	
		.		
3		MEND	{Fin de RDBUFF}	
4	WRBUFF	MACRO	&OUTDEV, &BUFADR, &RECLTH	
		.	{Versión estándar de SIC}	
		.		
5		MEND	{Fin de WRBUFF}	
		.		
		.		
6		MEND	{Fin de MACROS}	
			(a)	
1	MACROX	MACRO	{Define las macros para SIC/XE}	
2	RDBUFF	MACRO	&INDEV,&BUFADR,&RECLTH	
		.	{Versión SIC/XE}	
		.		
3		MEND	{Fin de RDBUFF}	
4	WRBUFF	MACRO	&OUTDEV,&BUFADR,&RECLTH	
		.	{Versión SIC/XE}	
		.		
5		MEND	{Fin de WRBUFF}	
		.		
		.		
6		MEND	{Fin de MACROX}	
			(b)	

FIGURA 4.3 Ejemplo de la definición de macros en el cuerpo de un macro.

Pueden ser útiles:

- La primera macro (MACROS) contiene proposiciones que definen RDBUFF, WRBUFF para SIC estándar
- (MACROX) define esas mismas macros para el sistema SIC/XE.
- El mismo programa puede ejecutarse sobre una máquina SIC estándar o una SIC/XE

Sintaxis

Nombre_de_macro: Es el nombre que le das a tu macro. Debes reemplazar esto con el nombre que desees para tu macro.

MACRO: Esta palabra clave indica que estás definiendo una macro.

parámetros: Puedes definir parámetros que tu macro puede aceptar entre paréntesis. Los parámetros son variables que se utilizan en el código de la macro y pueden ser reemplazados por valores cuando invocas la macro.

Tipos

Macros simples (o básicas): Las macros simples son bloques de código reutilizable que se utilizan para reemplazar secuencias de instrucciones repetitivas con una sola línea de código. Estas macros pueden o no tener parámetros y se utilizan para simplificar y abreviar el código fuente.

Macros avanzadas (o condicionales): Las macros avanzadas son bloques de código reutilizable que ofrecen una mayor flexibilidad y complejidad que las macros simples. Pueden incluir lógica condicional, bucles y otros tipos de estructuras de control, lo que las hace más adecuadas para la automatización de tareas específicas o para la creación de construcciones de programación más complejas.

Conclusión

Las macros son una parte esencial de la programación en lenguaje ensamblador y permiten a los programadores crear bloques de código reutilizable, mejorando la legibilidad del código y automatizando tareas repetitivas. Estas macros se utilizan para simplificar la escritura de programas, ya que permiten definir secuencias de instrucciones complejas en una forma más concisa y legible.

En el contexto de la programación en ensamblador, las macros se dividen principalmente en dos tipos: macros simples (o básicas) y macros avanzadas (o condicionales). Las macros simples son bloques de código que reemplazan secuencias de instrucciones repetitivas y pueden o no aceptar parámetros. Por otro lado, las macros avanzadas ofrecen una mayor flexibilidad y complejidad, lo que les permite incluir lógica condicional, bucles y otras estructuras de control.

El procesador de macros es una herramienta fundamental que se utiliza para definir, invocar y expandir macros en el código fuente. Cuando se invoca una macro en un programa, el procesador de macros reemplaza la llamada a la macro con el código real de la misma, un proceso conocido como "macroexpansión". Esto facilita la escritura de programas complejos y reduce la redundancia de código.

Es importante destacar que el diseño de un procesador de macros no está directamente relacionado con la estructura de la computadora en la que se ejecutará el programa. Los procesadores de macros se utilizan no solo en programación en lenguaje ensamblador, sino también en lenguajes de programación de alto nivel para automatizar tareas y mejorar la organización del código.

En resumen, las macros y los procesadores de macros son herramientas esenciales en la programación en ensamblador y en otros lenguajes. Permiten una programación más eficiente, legible y reutilizable al simplificar tareas comunes y reducir la redundancia en el código fuente.