El Espacio de Configuración de Dispositivos PCI (Desarrollo de Rutinas de Acceso)

Ing. Israel Rivera Zárate.
Profesor e Investigador del CIDETEC-IPN

Este documento es continuación del artículo titulado "Diseño de una Interfaz PCI para una Tarjeta Coprocesadora Basada en el DSP TMS320C40-40", aparecido en polibits en su número 21. Debido a esto, se da por entendido que el lector debe tomar dicho artículo como referencia base para la comprensión de este documento.

INTRODUCCIÓN

Cuando el equipo de computo se energiza, el software de configuración realiza un recorrido por el bus PCI para determinar la existencia de dispositivos así como de los recursos que estos requieren para su operación. Este proceso es comunmente referido como «*proceso de descubrimiento*». El programa que realiza esta función es referido frecuentemente como el «*enumerador del bus PCI*» [1].

Con el objetivo de facilitar este proceso, todos los dispositivos PCI deben implementar un conjunto básico de registros de configuración definidos por la especificación PCI. Dependiendo de sus características operativas, un dispositivo puede implementar registros opcionales de configuración, definidos también por la especificación.

El software de configuración efectua una serie de accesos a registros específicos para establecer la presencia y tipo de los dispositivos instalados en el bus PCI. Una vez determinada la presencia del dispositivo, el software accesa otros registros de configuración para establecer la cantidad de bloques de memoria y puertos de E/S que éste requiere para su operación. Entonces programa los decodificadores de direcciones de memoria y puertos de E/S implementados en el dispositivo para que responda en futuros accesos a estos rangos. Los rangos entre dispositivos son garantizados a ser mutuamente excluyentes.

Si el dispositivo manifiesta la necesidad de uso de una interrupción PCI (vía uno de sus registros de configuración), el software de configuración lo programa con la información de ruteo correspondiente indicándole cuál línea solicitud de interrupción del sistema (IRQ) le es asignada.

Si el dispositivo tiene capacidad de volverse maestro del bus, «*bus mastering*», el software de configuración puede accesar dos registros de configuración que permiten establecer la frecuencia con que solicitará la poseción del bus PCI y la duración promedio de la transferencia . El software de configuración emplea esta información para programar los registros de temporización de latencia del bus PCI implementados en el árbitro de bus PCI normalmente integrado en el ChipSet PCI y de esta manera garantizar un uso óptimo del bus.

TRES ESPACIOS DE DIRECCIONES : PUERTOS DE E/S, MEMORIA Y CONFIGURACIÓN

Los procesadores Intel (sólo por referirnos a la plataforma de PC), poseen la habilidad para direccionar dos espacios distintos: puertos de E/S y memoria. Los dispositivos PCI con capacidad de ser maestro del bus (incluyendo el mismo ChipSet PCI y los puentes PCI), desarrollan ciclos de acceso PCI de E/S y memoria para accesar las regiones de E/S y memoria de dispositivos PCI respectivamente. Además un tercer tipo de acceso, «*el ciclo de configuración*», es utilizado para accesar los registros de configuración.

Los ciclos de configuración se originan como un acceso ordinario del procesador anfitrión a las direcciones de puertos: CF8h y CFCh respectivamente. Estas direcciones corresponden con registros de operación del controlador de bus del sistema implementado en el ChipSet PCI. El registro ubicado en CF8h es frecuentemente denominado CONFADDRESS

y traduce el dato enviado en una dirección dentro del espacio de configuración. El registro ubicado en CFCh es frecuentemente denominado como CONFDATA y almacena el dato a transferir a la región de configuración defina por CONFADDRESS. Rutinas dispuestas en el BIOS PCI agilizan este procedimeinto permitiendo el acceso libre y transparente [2].

El espacio de configuración PCI es dividido y repartido entre cada dispositivo funcional contenido dentro de un único dispositivo físico (circuito integrado). Las primeras 16 doublewords (1 doubleword=32 bits), son referidas como el encabezado de configuración, *«configuration header»*. El formato y uso de esta área es definida por la especificación PCI. Actualmente existen dos tipos básicos de encabezados de configuración: encabezado tipo 0 para todos los dispositivos, diferentes a los puentes Host/PCI, PCI/PCI, o PCI/ISA, los cuales emplean el tipo 1.

Encabezado de configuración tipo 0

Como se mencionó en el párrafo anterior, el encabezado tipo 1 es más bien asignado a los dipositivos de soporte del sistema de cómputo. Tomándo como base que nuestro interés se centra en el desarrollo de aplicaciones orientadas el bus PCI, parece más conveniente describir la arquitectura del encabezado de configuración tipo 0. Ver **figura 1**.

La descripción formal respecto al significado y uso de los registros descritos en el encabezado de configuración escapa a los propósitos del artículo por lo que tan solo se indican brevemente en la **tabla 1**.

31 24	23 16	15 8	7 0	
DEVICE ID			VENDOR ID	00
STATUS			COMMAND	04
CLASS CODE			REV ID	08
BIST	HEADER TYPE	LATENCY TIMER	CACHE L Z	0C
BASE ADDRESS REGISTER 0				10
BASE ADDRESS REGISTER 1				14
BASE ADDRESS REGISTER 2				18
BASE ADDRESS REGISTER 3				1C
BASE ADDRESS REGISTER 4				20
BASE ADDRESS REGISTER 5				24
RESERVED = 0				28
RESERVED = 0				2C
EXPANSION ROM BASE ADDRESS				30
RESERVED = 0				34
RESERVED = 0				38
MAX_LAT	MIN_GNT	INTERRUPT PIN	INTERRUPT LINE	3C

Figura 1. Encabezado tipo 0 del espacio de configuración de dispositivos PCI.

Plug and Play

Los dispositivos PCI forman parte de la arquitectura de la especificación Plug and Play (también llamada PnP). PnP fue desarrollada por Microsoft con la cooperación de Intel y muchos otros fabricantes de hardware. Como su nombre sugiere, la meta de Plug and Play es crear una máquina cuyo hardware y software trabajen conjuntamente para configurar dispositivos y asignar recursos automáticamente, de tal forma que al instalar un nuevo dispositivo pueda utilizarse inmediatamente sin mayor necesidad de configuración.

La detección automática y configuración de dispositivos no es una tarea fácil, para realizarse, requiere la cooperación de diferentes componentes hardware y software: hardware del sistema, hardware de periféricos, BIOS del sistema y sistema operativo [3].

Hardware del Sistema. El hardware del sistema a través del Chipset del sistema y los controladores del bus del sistema deben ser capaces de manejar dispositivos PnP. Los sistemas modernos basados en PCI han sido diseñados con PnP en mente.

Hardware de Periféricos. Los dispositivos que se añaden al sistema deben ser compatibles con PnP.

BIOS. El BIOS del sistema juega un papel clave, ya que las rutinas contenidas en él realizan la tarea de recolectar la información de los diferentes dispositivos instalados, así como los recursos que requieren. El BIOS comunica esta información al sistema operativo, que la utiliza para configurar sus drivers y otros programas para permitir que los dispositivos trabajen correctamente.

Desplazamiento	Abreviación	Descripción
00h-01h	VID	Identificación del fabricante del dispositivo.
02h-03h	DID	Identificación del dispositivo.
04h-05h	PCICMD	Comando de acceso PCI (acceso de lectura o escritura).
06h-07h	PCISTS	Información del estado del acceso.
08h	RID	Identificación del número de revisión del fabricante.
09h-0Bh	CLCD	Tipo de dispositivo: Clase base, Subclase, etc.
0Ch	CALN	Tamaño de la linea de cache en doublewords.
0Dh	LAT	Tiempo promedio en el que el dispositivo será maestro del bus.
0Eh	HDR	Tipo de header =0.
0Fh	BIST	Auto Prueba («Build-In Self Test»). Permite la implementación de rutinas de diagnóstico del dispositivo
10h-27h	BADR0-BADR5	Registros de dirección base. En estos registros se definen las regiones de memoria o puertos de E/S y sus tamaños.
28h-2Fh	——	Reservado.
30h	EXROM	Provee un mecanismo para la asignación de espacio en memoria física de una ROM de expansión
34h-3Bh	——	Reservado.
3Ch	INTLN	Línea IRQ del sistema asignada al dispositivo.
3Dh	INTPIN	Identifica la interrupción PCI del sispositivo (/INTA, /INTB, /INTC o /INTD).
3Eh	MINGNT	Duración promedio de un burst en modo maestro.
3Fh	MAXLAT	Frecuencia promedio de uso del bus PCI en modo maestro.

Tabla 1. Descripción de los registros del encabezadode configuración tipo 0.

Sistema Operativo. Finalmente, el sistema operativo debe estar diseñado para trabajar con el BIOS (y por lo tanto indirectamente con el hardware). El sistema operativo da de alta el software de bajo nivel (tal como device drivers) que sea necesario para el dispositivo que ha de ser utilizado por las aplicaciones. Además se comunica con el usuario, notificando los cambios en la configuración, y permite modificar la asignación de recursos si fuese necesario.

La mayor parte del trabajo que permite a Plug and Play operar correctamente la realiza el BIOS del sistema durante el proceso de arranque (boot). Si el BIOS tuviera que asignar recursos a cada dispositivo PnP en cada arranque del sistema, resultarían dos problemas. El primero, invertiría tiempo en hacer algo que acaba de realizar anteriormente, en cada arranque, sin ningún propósito. Después de todo, la mayoría de la gente no cambia el hardware del sistema constantemente. Segundo, y más importante, es posible que el BIOS no realice siempre la misma decisión cuando decide asignar recursos, pudiendo realizar modificaciones incluso cuando el hardware no ha sido cambiado.

Por ello se diseñó el ESCD (Extended System Configuration Data), que es un área especial que forma parte de la memoria CMOS donde son resguardados los parámetros del BIOS. Esta área de memoria es utilizada para mantener información de configuración del hardware del sistema. Al momento del arranque el BIOS verifica esta área de memoria y si no detecta cambio alguno respecto al último arranque, asume que no requiere configurar nada y brinca esa porción del proceso de arranque.

El ESCD es utilizado también como un enlace de comunicación entre el BIOS y el sistema operativo. Ambos utilizan el área ESCD para leer el estado actual del hardware y grabar los cambios. Windows 95 lee el ESCD para determinar si hay algún cambio en el hardware y reaccionar correspondientemente. Windows 95 permite a los usuarios modificar las asignaciones de recursos realizadas por PnP, manualmente a través del Administrador de dispositivos. Esta información es grabada en el área ESCD, así que el BIOS conocerá los cambios realizados en el arranque siguiente y no intentará modificar las asignaciones.

Propósito del BIOS PCI

El sistema operativo (excepto por el micro kernel específico de la plataforma), programas de aplicación y device drivers no son capaces de accesar directamente los registros de configuración de dispositivos PCI. Los métodos hardware utilizados para implementar esta capacidad son específicos de la plataforma. Cualquier software que accese directamente estos mecanismos es por lo tanto, por definición, específico de la plataforma. Esto conduce a problemas de compatibilidad, esto es, el software trabaja en algunas plataformas pero no en otras.

En vez de esto, las solicitudes de comunicación deben dirigirse al BIOS PCI. El BIOS es específico de la pla-

taforma. Es implementado en firmware y posiblemente en la capa de abstracción del hardware HAL (Hardware Abstarcción Layer) del sistema operativo [2]. El BIOS provee los siguientes servicios:

- Permite la determinación de los mecanismos de configuración soportados por el chipset PCI.
- Permite la determinación del rango de los buses PCI presentes en el sistema.
- Busca todas las instancias de un dispositivo PCI específico o un dispositivo correspondiente a una clase específica.
- Permite la lectura y escritura de los registros de configuración de dispositivos PCI

Soporte de entornos de sistemas operativos

Diferentes sistemas operativos poseen diferentes características operacionales (tales como el método que define el uso de la memoria del sistema, el método utilizado para invocar los servicios del BIOS). En sistemas basados en la familia de procesadores x86, el sistema operativo que se ejecuta en una plataforma particular cae en una de las siguientes categorías:

- Sistema operativo en modo real (en otras palabras MS-DOS).
- Modo protegido 286.
- Modo protegido 386 (modelo segmentado y modelo plano).

Modo real

Los sistemas operativos en modo real, tal como MS-DOS, están escritos para ser ejecutados por el procesador 8088. Ese procesador es capaz de direccionar únicamente 1MB de la memoria del sistema (00000h a FFFFFh). MS-DOS realiza llamadas al BIOS PCI mediante la ejecución de la interrupción por software 1Ah. Para especificar un servicio particular deben cargarse previamente parámetros en los registros de segmento del procesador. Ver **tabla 2**.

FUNCION	Valor en AH	Valor en AL
Identificador de funciones PCI	B1h	
Prueba la existencia del BIOS PCI		01h
Encuentra un dispositivo PCI		02h
Encuentra un dispositivo PCI según su clase de código		03h
Genera ciclo especial		06h
Lee un byte de un registro de configuración		08h
Lee una word de un registro de configuración		09h
Lee una doubleword de un registro de configuración		0Ah
Escribe un byte a un registro de configuración		0Bh
Escribe una word a un registro de configuración		0Ch
Escribe una doubleword a un registro de configuración		0Dh
Obtiene las opciones de ruteo de interrupciones PCI		0Eh
Asigna una interrupción PCI		0Fh

Tabla 2. Servicios del BIOS PCI.

Rutinas de acceso al espacio de configuración

Tomando como base los servicios a interrupción para acceso al espacio de configuración dispuestos en el BIOS PCI se desarrollaron un conjunto de rutinas básicas fundamentadas en [4] y [5]; que permiten el acceso a los registros de configuración de cualquier dispositivo instalado en el bus PCI. Estas rutinas se presentan en las columnas siguientes:

Fragmento de código ejemplo

Contando con las rutinas básicas de acceso a registros de configuración se desarrolló un programa de prueba en el que se realizan únicamente operaciones de lectura a todos los registros de configuración de acuerdo con la arquitectura de encabezados tipo 0 mostrada en la **figura 1**.

```
;*************************************************************************
;      llamada a funciones del BIOS PCI para leer o escribir
;      a registros de configuración
;*************************************************************************
;                    READ_C_BYTE
;      Lee un registro de  configuración de 8 bits
;   DI: número de registro
;   CL: byte leído
;   Bandera Carry: 1=error, 0=exitoso.
;-------------------------------------------------------------------------
read_c_byte proc   near
   push    bx                          ;salva valores de registros
   push    ax                          ;
   mov     bh, BUS_NUM                 ;carga número de bus
   mov     bl, DEV_FUNC                ;carga número de dispositivo
           ; y numero de función
   mov     ah, PCI_FUNC_ID             ;carga identificador de función PCI
   mov     al, READ_CONFIG_BYTE ;carga servicio de lectura de byte
   int     1ah                         ;interrupción del bios
   pop     ax                          ;
   pop     bx                          ;restablece valor a registros
   jnc     good                       ;si tuvo exito regresa a la
                                       ;rutina principal
good:   ret                            ;regreso a la rutina principal
read_c_byte endp
;-------------------------------------------------------------------------
;                    READ_C_WORD
;
;      Lee un registro de configuración de 16 bits
;   DI: número de registro
;   CX: word leída
;   Bandera Carry: 1=error, 0=exitoso.
;-------------------------------------------------------------------------
read_c_word proc near
   push    bx                          ;salva registros
   push    ax                          ;
   mov     bh, BUS_NUM                 ;
   mov     bl, DEV_FUNC                ; carga número de dispositivo
                                       ; y número de función
   mov     ah, PCI_FUNC_ID             ;carga identificador de función PCI
   mov     al, READ_CONFIG_WORD        ;carga servicio de lectura de word
   int     1ah                         ;interrupción del bios
   pop     ax                          ;restablece valores a registros
   pop     bx                          ;
   ret
read_c_word endp
;-------------------------------------------------------------------------
;                    READ_C_DWORD
;
;      Lee un registro de configuración de 32 bits
;   DI: número de registro
;   ECX: byte leído
;   Bandera Carry: 1=error, 0=exitoso.

;-------------------------------------------------------------------------
read_c_dword proc near
   push    bx                          ;salva registros
   push    ax                          ;
   mov     bh, BUS_NUM                 ;
   mov     bl, DEV_FUNC                ;carga número de dispositivo y
                                       ;número de función
   mov     ah, PCI_FUNC_ID             ;carga identificador de función PCI
   mov     al, READ_CONFIG_DWORD;carga servicio lee double word
   int     1ah                         ;interrupción del bios
   pop     ax                          ;restablece valores a registros
   pop     bx                          ;
   ret
read_c_dword endp
;-------------------------------------------------------------------------
;                    WRITE_C_BYTE
;
;      Escribe un dato de 8 bits a un registro de configuración
;   DI: número de registro
;   CL: valor a escribir
;   Bandera Carry: 1=error, 0=exitoso.
;-------------------------------------------------------------------------
write_c_byte proc near
   push    ax                          ;salva valor de registros
   push    bx                          ;
   mov     ah, PCI_FUNC_ID             ;carga identificador de función PCI
   mov     al, WRITE_CONFIG_BYTE       ;servicio de ecritura de byte
   mov     bh, BUS_NUM                 ;carga número de bus
   mov     bl, DEV_FUNC                ;carga número  de función y
                                       ;dispositivo
   int     1ah                         ;interrupción del bios
   pop     bx                          ;restablece registros
   pop     ax                          ;
   ret
write_c_byte endp
;-------------------------------------------------------------------------
;                    WRITE_C_WORD
;
;      Escribe un dato de 16 bits a un registro de configuration
;   DI: número de registro
;   CX: valor a escribir
;   Bandera Carry: 1=error, 0=exitoso.
;-------------------------------------------------------------------------
write_c_word proc near
   push    bx                          ;save registers
   push    ax                          ;
   mov     ah, PCI_FUNC_ID             ;load function id
   mov     al, WRITE_CONFIG_WORD ;load write function value
   mov     bh, BUS_NUM                 ;load bus number
   mov     bl, DEV_FUNC                ;load device and function number
   int     1ah                         ;call bios
   pop     ax                          ;
   pop     bx                          ;restore register values
   ret                                 ;
write_c_word endp
;-------------------------------------------------------------------------
;                    WRITE_C_DWORD
;
;      Escribe un dato de 32 bits a un registro de configuración
;   DI: número de registro
;   ECX: valor a escribir
;   Bandera Carry: 1=error, 0=exitoso.
;-------------------------------------------------------------------------
write_c_dword proc near
   push    ax                          ;salva valor de registros
   push    bx                          ;
   mov     ah, PCI_FUNC_ID             ;es una función PCI
   mov     al, WRITE_CONFIG_DWORD  ;servicio de función PCI
   mov     bh, BUS_NUM                 ;numero de bus PCI
   mov     bl, DEV_FUNC                ;número de dispositivo y función
   int     1ah                         ;interrupción del BIOS
   pop     bx                          ;restablece valor de registros
   pop     ax                          ;
   ret
write_c_dword endp
;-------------------------------------------------------------------------
```

El fragmento asume que existen localidades reservadas donde han sido depositadas previamente las dirección de desplazamiento de cada registros particular. Asume también que se cuenta con una rutina de despliegue de información en pantalla.

BIBLIOGRAFÍA

[1] PCI System Architecture. Tom Shanley & Don Anderson. Mindshare

[2] Intel 430MX PCISET DataBook.

[3] http://www.pcguide.com

[4] S5933 PCI Controller DataBook

[5] S5933 PCI MatchMaker Developer's Kit, Technical Reference Manual.


```
;-------------------------------------------------------------------------
;       READ_ALL_CONFIG_REGS
;
;  Muestra el estado de todos los registros de configuracíon
;
;-------------------------------------------------------------------------
read_all_config_regs proc near
   push  ax                     ;resguarda registros
   push   dx
   push  cx
   movcl, 21d                   ;total de registros
   mov ah,00h                   ;número actual del registro
cregs:
   push  cx                     ;resguarda registros
   call  do_c_read_write        ;lee registro siguiente
   pop cx                       ;restablece el contador
   dec  cl                      ;decrementa el contador
   inc   al                     ;siguiente registro
   cmpcl, 0                     ;verifica si contador es cero
   jnz   cregs                  ;
   pop cx                       ;recupera valor de registros
   pop dx                       ;
   pop   ax
   ret
read_all_config_regs endp
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
;       DO_C_READ_WRITE
;
;  Lee todos los registros de configuración
;-------------------------------------------------------------------------
do_c_read_write  proc  near
   cmpah, 0                     ;
   jz rvid                      ;lee VID
   cmpah, 1                     ;
   jz rdid                      ;lee DID
   cmpah, 2                     ;
   jz rcmd                      ;lee CMD
   cmpah, 3                     ;
   jz rsts                      ;lee STS
   cmpah, 4                     ;
   jz rrid                      ;lee RID
   cmpah, 5                     ;
   jz rclcd                     ;lee CLCD
   cmpah, 6                     ;
   jz rcaln                     ;lee CALN
   cmpah, 7                     ;
   jz rlat                      ;lee LAT
   cmpah, 8                     ;
   jz rhdr                      ;lee HDR
   cmpah, 9                     ;
   jz rbist                     ;lee BIST
   cmpah, 10d                   ;
   jz rbadr0                    ;lee BADR0
   cmpah, 11d                   ;
   jz rbadr1                    ;lee BADR1
   cmpah, 12d                   ;
   jz rbadr2                    ;lee BADR2
   cmpah, 13d                   ;
   jz rbadr3                    ;lee BADR3
   cmpah, 14d                   ;
   jz rbadr4                    ;lee BADR4
   cmpah, 15d                   ;
   jz rbadr5                    ;lee BADR5
   cmpah, 16d                   ;
   jz rexrom                    ;lee EXROM
   cmpah, 17d                   ;
   jz rintln                    ;lee INTLN
   cmpah, 18d                   ;
   jz rintpin                   ;lee INTPIN
   cmpah, 19d                   ;
   jz rming                     ;lee MINGNT
   cmpah, 20d                   ;
   jz rmaxl                     ;lee MAXLAT
   jmp  gb_cread                ;
```

```
rvid: movdi, VID_ADDR          ;carga dirección de VID
   jmp word_reg                ;
rdid: movdi, DID_ADDR          ;carga dirección de DID
   jmp word_reg                ;
rcmd:  movdi, CMD_ADDR         ;carga dirección de COMMAND
   jmp word_reg                ;
rsts: movdi, STS_ADDR          ;carga dirección de STATUS
   jmp word_reg                ;
rrid: movdi, RID_ADDR          ;carga dirección de REVISION ID
   jmp byte_reg                ;
rclcd:
movdi, 0008h                   ;utiliza una localidad anterior a CLCD para
                               ;realizar lectura de 32 bits.
   jmp b3_reg                  ;
rcaln:
movdi, CALN_ADDR               ;carga dirección de CALN
   jmp byte_reg    ;
rlat: movdi, LAT_ADDR          ;carga dirección de LATENCY
   jmp byte_reg    ;
rhdr:   movdi, HDR_ADDR        ;carga dirección de HEADER
   jmp byte_reg                ;
rbist:
movdi, BIST_ADDR               ;carga dirección de BIST
   jmp byte_reg                ;
rbadr0:
movdi, BADR0_ADDR              ;carga dirección de BADR0
jmp dword_reg                  ;
rbadr1:
movdi, BADR1_ADDR              ;carga dirección de BADR1
   jmp dword_reg               ;
rbadr2:
movdi, BADR2_ADDR              ;carga dirección de BADR2
   jmp dword_reg               ;
rbadr3:
movdi, BADR3_ADDR              ;carga dirección de BADR3
   jmp dword_reg               ;
rbadr4:
movdi, BADR4_ADDR              ;carga dirección de BADR4
   jmp dword_reg               ;
rbadr5:
movdi, BADR5_ADDR              ;carga dirección de BADR5
```

```
   jmp dword_reg               ;
rexrom:
movdi, EXROM_ADDR              ;carga dirección de EXROM
   jmp dword_reg               ;
rintln:
movdi, INTLN_ADDR              ;carga dirección de INTLN
   jmp byte_reg                ;
rintpin:
movdi, INTPIN_ADDR             ;carga dirección de INTPIN
   jmp byte_reg                ;
rming:
movdi, MINGNT_ADDR             ;carga dirección de MINGNT
   jmp byte_reg                ;
rmaxl:
movdi, MAXLAT_ADDR             ;carga dirección de MAXLAT
   jmp byte_reg                ;
byte_reg:                      ;lee registro de 8 bits
   call read_c_byte            ;lee el registro
   movBYTE_VAL, cl             ;valor a imprimir en pantalla
   call print_byte             ;lo presenta en pantalla
   jmp gb_cread                ;regresa de la rutina
word_reg:                      ;lee registro de 16 bits
   call read_c_word            ;lee el registro
   movWORD_VAL, cx             ;valor listoa presentar en pantalla
   call print_word             ;lo presenta en pantalla
   jmp gb_cread                ;regreso
b3_reg:                        ;operación de 24 bits (CLCD)
   call read_c_dword           ;lee 24 bitscomo un registro de 32 bits
   movDWORD_VAL, ecx           ;listo a presentar en pantalla
   call print_3bytes           ;imprime en pantalla
   jmp gb_cread                ;regreso
dword_reg:                     ;operación de 32 bits
   call read_c_dword           ;lee registro
   movDWORD_VAL, ecx           ;listo a presentar en pantalla
   call print_dword            ;imprime en pantalla
   jmp gb_cread                ;regreso
gb_cread:
   ret
do_c_read_write  endp
;--------------------------------------------------------------------------
-
```