Interrupciones en 8086

Departamento de Arquitectura[1]

[1]Instituto de Computación
Facultad de Ingeniería
Universidad de la República

Arquitectura de Computadoras, 2017

Contextualización

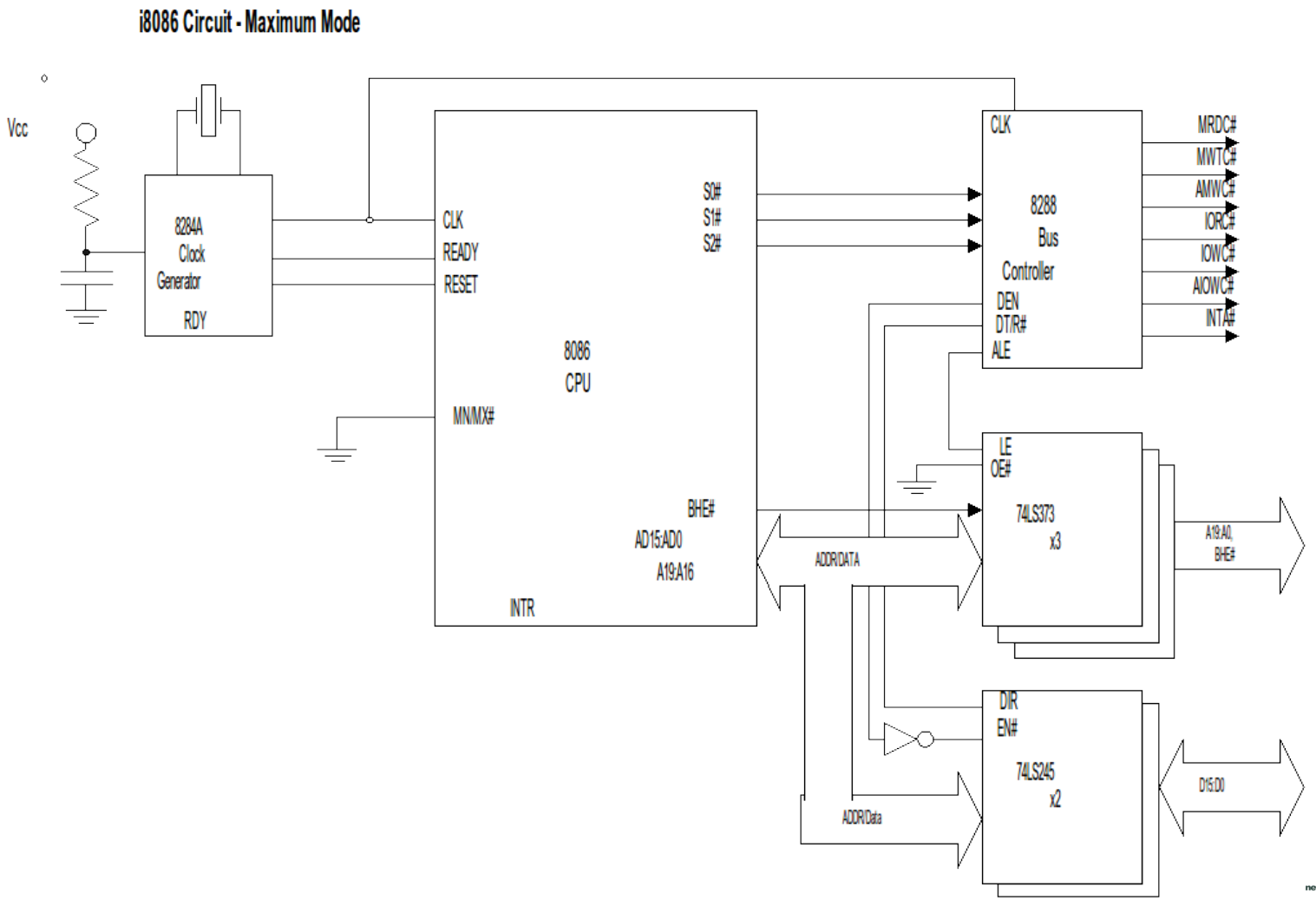
- Motivación de las interrupciones.
- Pedido, detección y atención.
- Controlador de interrupciones.
- Aspectos generales de 8086.
- E/S e interrupciones.

Contenido

Construyendo un sistema basado en el 8086/8

- Los microprocesadores 8086/8 necesitan circuitos extra para construir un sistema.
- Los buses de datos y direcciones se multiplexan en los mismos pines del procesador. Se necesita lógica extra para demultiplexar direcciones y datos, y poder acceder a RAMs y ROMs.
- Modos de funcionamiento (Máximo y Mínimo).
- El Modo Máximo el 8086/8 necesita al menos los siguientes circuitos extra: 8288 Bus Controller, 8284A Clock Generator, 74HC373s y 74HC245s.

Circuitos en modo máximo



Evolución: 80186/80188

- Set de Instrucciones aumentado
- Componentes del sistema "on-chip" (SOC)
  - Clock generator
  - Controlador DMA
  - Controlador interrupciones
  - Timer
  - etc...
- No utilizado en PCs
- Popular en sistemas embebidos

Señal RESET

- RESET es activa en nivel bajo.
- Pone al 8086/8 en un estado definido
  - Limpia los registros de flags, segmento, etc.
  - Pone la dirección de programa efectiva en 0xFFFF0 (CS = 0xF000 e IP = 0xFFF0)
- Programas en el 8086/8 siempre arrancan en FFFF0H después de un Reset
- En esta dirección deben instalarse las rutinas de inicialización del sistema: en el PC, la ROM BIOS
- Esta característica se mantiene en las últimas generaciones de procesadores

Direccionamiento de memoria y E/S

- Los procesadores Intel tienen el espacio de direccionamiento de E/S separado de la memoria principal.
- Uso de las señales IOR# y IOW#
- Se corresponden con instrucciones separadas para acceder la E/S y la memoria
  - MOV AL, [BX] ; acceso a memoria
  - IN AL, 2Ch ; acceso a E/S
- Algunos procesadores tienen un espacio de direcciones unificado. Los dispositivos de E/S son decodificados en el mapa de memoria principal ("E/S mapeada en memoria")

Direccionamiento de memoria y E/S

Comparación

- Ventajas de la E/S mapeada en memoria
  - Dispositivos de E/S accedidos por instrucciones normales - no se necesitan instrucciones separadas
  - Se reduce el tamaño del set de instrucciones
  - No se necesitan pines especiales (IOR# y IOW#)
- Ventajas de espacios de direccionamiento separados
  - Todo el mapa de direcciones disponible para memoria
  - La E/S puede usar instrucciones más pequeñas y rápidas
  - Fácil distinguir accesos de E/S en lenguaje ensamblador
  - Menos hardware para decodificar E/S.

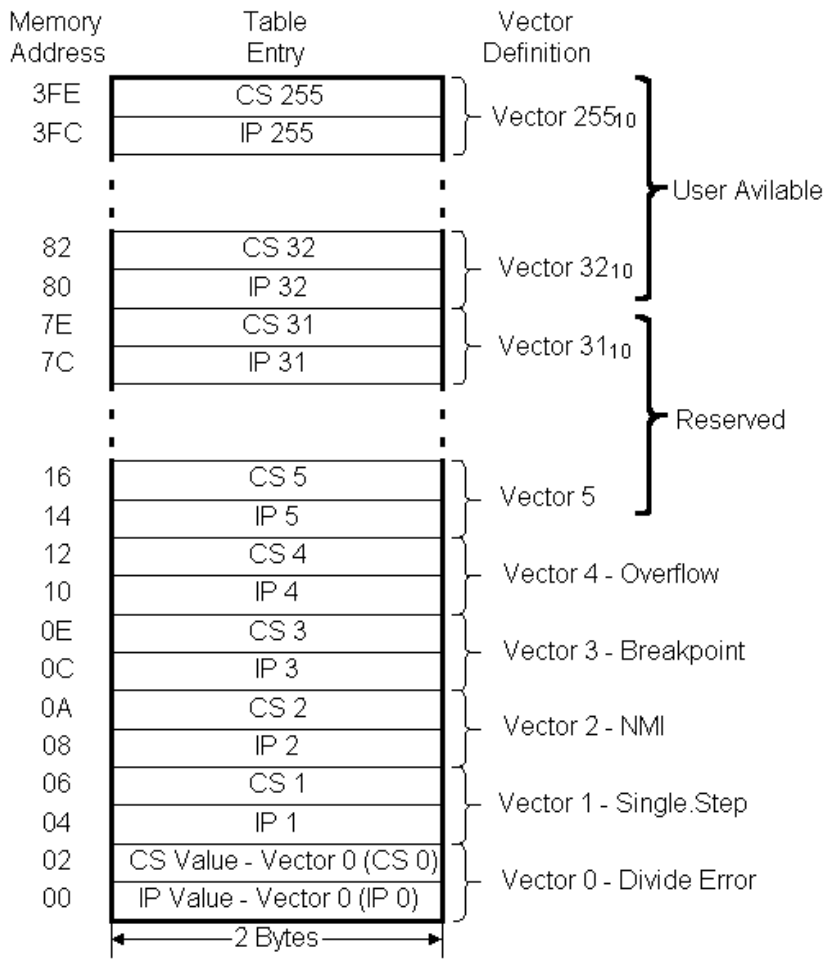
Tipos de interrupciones

- Tipos
  - Hardware: dispositivos de entrada salida
  - Internas: división entre cero (instrucción DIV)
  - Software: llamadas al sistema (instrucción INT)
  - No enmascarables.
- Cada interrupción lleva asociado un número que identifica al servicio que se debe invocar.

Vector de interrupciones

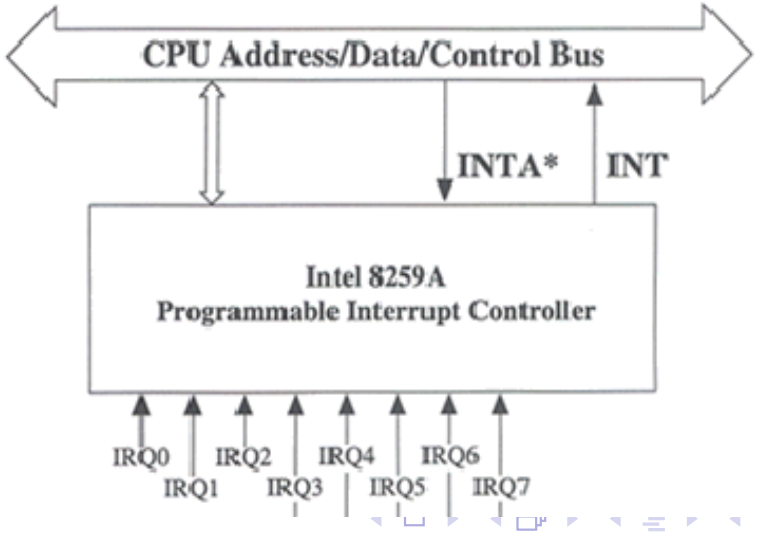
- Las localizaciones de memoria 00000H a 003FFH están reservadas para el vector de interrupciones.
- Existen 256 tipos posibles de interrupciones
- Cada Handler o Rutina de Servicio de Interrupción está direccionada por un puntero de 4 bytes: 16 bits de segmento y 16 bits de offset
- Las rutinas y los punteros deben instalarse antes de habilitar las interrupciones
  - Servicios de la BIOS
  - Servicios del Sistema Operativo

Vector de interrupciones



Controlador 8259

- Un dispositivo genera una señal de interrupción
- Si no hay otra interrupción de mayor prioridad ejecutándose o pendiente, el 8259 envía la señal INTR al 8086
- Si IF está a 1, el 8086 acepta la interrupción enviando una señal INTA al 8259
- El 8259 coloca en el bus de datos el identificador del dispositivo elegido



Interrupciones enmascarables y no enmascarables

- Las interrupciones pueden enmascararse globalmente usando la bandera Interrupt Enable (IF o I)
- IF es seteada por la instrucción STI y reseteada mediante CLI
- Interrupciones no enmascarables (Non Maskable Interrupts-NMI) son prioritarias y como su nombre indica NO se pueden enmascarar
- Uso de la NMI
  - Parity Error
  - Power fail
  - Etc

Assembler

- Operadores
  - Operador offset y seg
  - Formato: oper etiqueta—variable
- Instrucciones
  - INT
  - CLI y STI
  - IRET

Instrucción INT

Formato: INT op_1

Tipo Args: (i)

Lógica:

PUSHF
IF=0
TF=0
CALL FAR tabla_int(op_1)

Descripción: Genera una interrupción software tipo op_1.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	x	x	-	-	-	-	-

Instrucción CLI

Formato: CLI

Lógica: IF := 0

Descripción: Borra la bandera de activación de interrupciones (IF) y desactiva las interrupciones enmascarables (las que aparecen sobre la linea INTR del procesador).

- Las interrupciones enmascarables se pueden activar o desactivar.
- Las interrupciones no enmascarables (las que aparecen sobre la linea NMI) no se pueden desactivar.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
-	-	0	-	-	-	-	-	-

Instrucción IRET

Formato: IRET

Lógica:

RET
POPF

Descripción: Retorna de una interrupción. Funciona en forma equivalente al RET utilizado para retornar de una llamada CALL con la diferencia que esta instrucción restaura las flags del stack antes de retornar.

Banderas:

OF	DF	IF	TF	SF	ZF	AF	PF	CF
X	x	x	x	x	x	x	x	x

Habilitación de interrupciones

- Instrucción STI
- IRET
  - Restaura CS:IP desde el stack
  - Restaura el registro de banderas desde el stack
- ¿Quién habilita interrupciones?

Instrucción IN

Formato: IN op_1, op_2

Tipo Args: (AL, AX) (i,DX)

Lógica:

si $op_1 = \mathbf{AL}$
$\mathbf{AL} = I/O(op_2)$
sino si $op_1 = \mathbf{AX}$
$\mathbf{Ax} = I/O(op_2)$
fin si

Descripción: Transfiere un byte o una palabra de una puerta de entrada del procesador al registro Al o Ax, respectivamente.

El nro. de la puerta se puede especificar mediante:

- Un valor fijo (de o a 255).
- Un valor variable, el contenido en el registro Dx (de 0 a 65535), pudiéndose acceder a 64K puertas de entrada.

Instrucción OUT

Formato:	OUT	op_1,	op_2
Tipo Args:		(i,DX)	(Al, Ax)

Lógica:

si $op_1 = \mathbf{Al}$
    $I/O(op_2) = \mathbf{Al}$
sino si $op_1 = \mathbf{Ax}$
    $I/O(op_2) = \mathbf{Ax}$
fin si

Descripción: Transfiere un byte o una palabra desde el registro Al o Ax a una puerta de salida del procesador

El nro. de la puerta se puede especificar mediante:

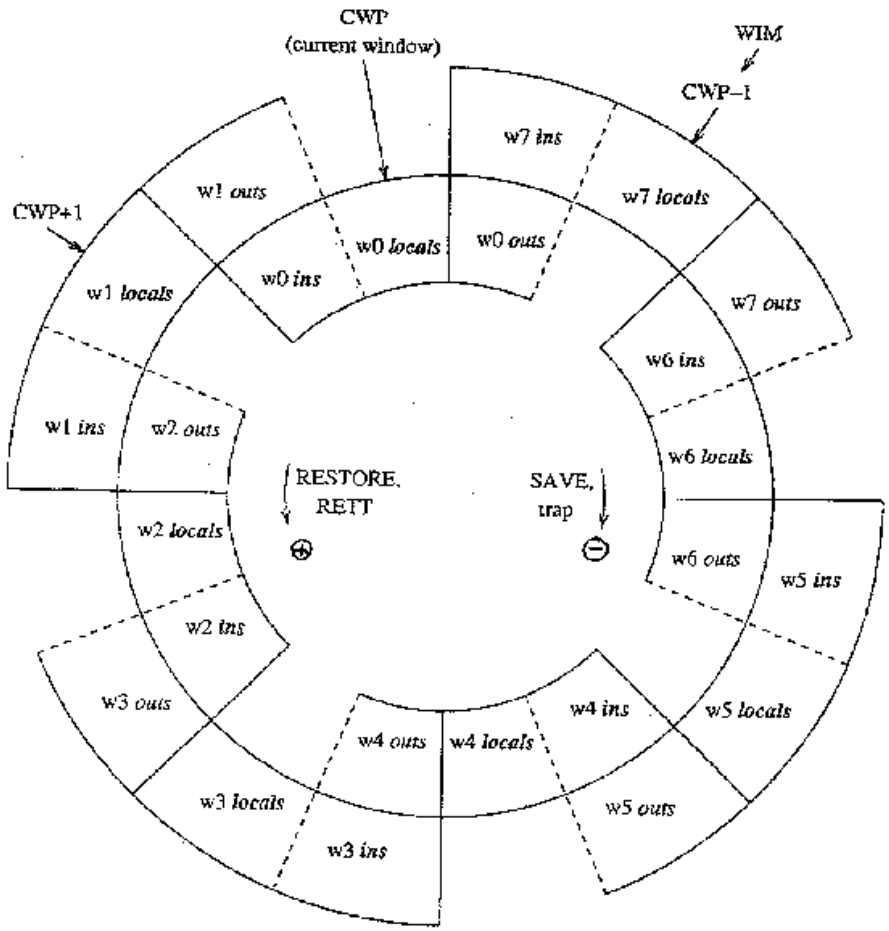
- Un valor fijo (de o a 255).
- Un valor variable, el contenido en el registro Dx (de 0 a 65535), pudiéndose acceder a 64K puertas de salida.

Resumen

- Acciones tomadas por el microprocesador
  - Salva las banderas en el stack
  - Deshabilita interrupciones
  - Salva CS:IP actual en el stack
  - Salta a la rutina de atención requerida
- Debe tenerse en cuenta que
  - Debo salvar el contexto
  - Cuidado con el stack
  - El vector de interrupciones ocupa desde la dirección de memoria absoluta 0 a la 1023.
  - Cada elemento del vector de interrupciones es una dirección segmentada que indica donde se encuentra el código del manejador de la interrupción.

Preservar el contexto

- 8086, stack.
- SPARC, ventanas.



Realidad

- Se desea controlar un LED conectado al bit menos significativo del byte de ES sólo escritura ES_LED, de modo que permanezca por siempre un segundo prendido y otro segundo apagado.
- Notas:
  - El procesador no está dedicado a esta aplicación.
  - La interrupción de TIMER es la 08h (elemento de índice 8 dentro del vector de interrupciones), y se ejecuta 18 veces por segundo.

Alto nivel

```
var
  cantTics: integer;
  estadoLED: byte;
procedure principal()
begin
  cantTics:=0;
  estadoLED:=0;
  out(ES_LED,estadoLED);
  {deshabilitoInterrupciones}
  {instaloManejadorTimer}
  {habilitoInterrupciones}
end;
```

```
procedure TIMER();
begin
  if (cantTics=18) then
    begin
      cantTics:=0;
      estadoLED:=notb(estadoLED);
      out(ES_LED,estadoLED);
    end
  else
    cantTics:=cantTics+1;
end
```

Assembler

```
ES_LED equ …

cantTics db 0
estadoLED db 0

principal proc
  mov byte ptr CS:[cantTics],0 ;cantTics:=0
  mov byte ptr CS:[estadoLED],0 ;estadoLED:=0
  mov dx,ES_LED
  xor al,al
  out dx,al   ; out(ES_LED,estadoLED)
  xor ax,ax
  mov es,ax
  cli           ; {deshabilitoInterrupciones}
  ;instaloManejadorTimer sin llamar al
  ;manejador actual
  ;indico desplazamiento y segmento
  mov word ptr es:[8*4],offset TIMER
  mov  word ptr es:[8*4+2],cs
  sti ; {habilitoInterrupciones}
  ret
principal endp
```

```
TIMER proc far
  push ax
  push bx
  push dx
  mov bl,cs:[cantTics]
  cmp bl,18
  jne else
  mov bl,0
  mov al, cs:[estadoLED]
  not al
  mov cs:[estadoLED],al
  mov dx,ES_LED
  out dx,al
  jmp fin
else:
  inc bl
fin:
  mov cs:[cantTics],bl
  pop dx
  pop bx
  pop ax
  iret
```

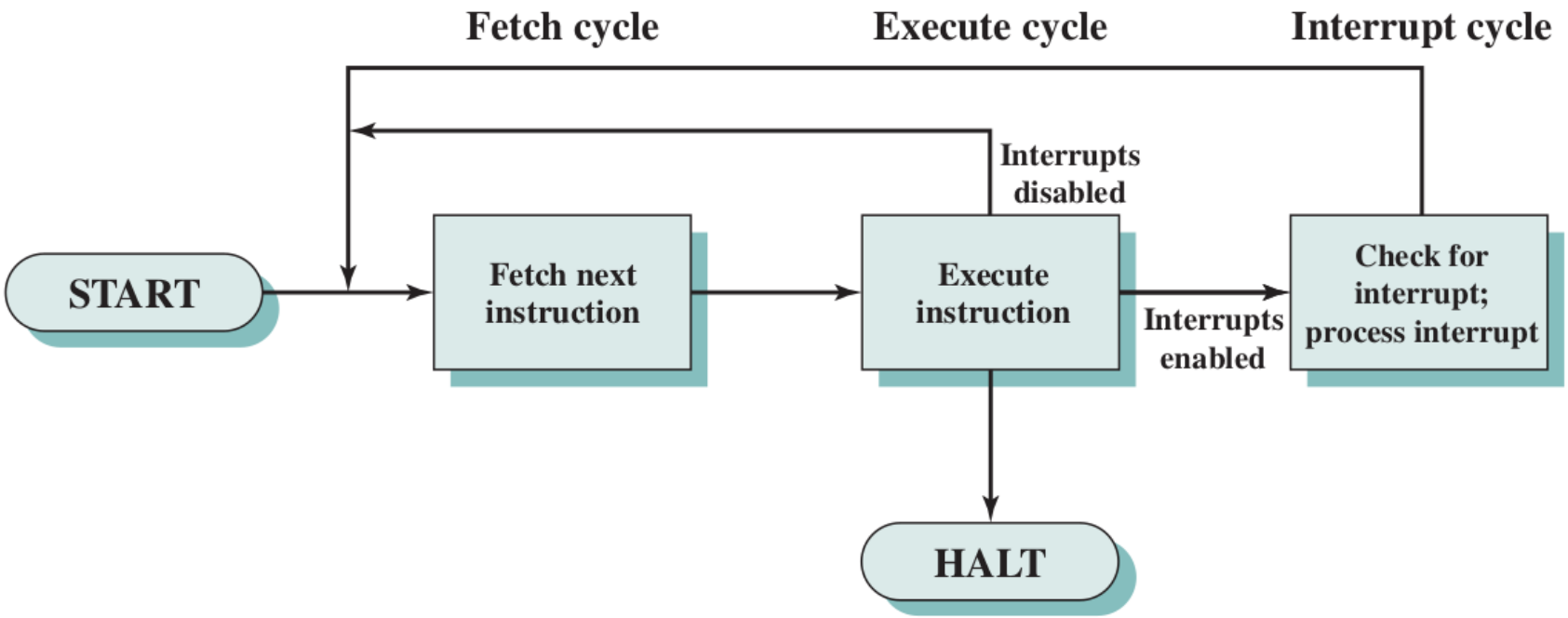
Controlador de interrupciones

- 8259.
- Registros:
  - Definir identificadores para cada IRQ.
  - Prioridades.
  - Flanco o nivel.

Controlador de interrupciones
Alternativas a INT/INTA

- Múltiples líneas de interrupción.
- Múltiples líneas de interrupción codificadas.

Identificación por software



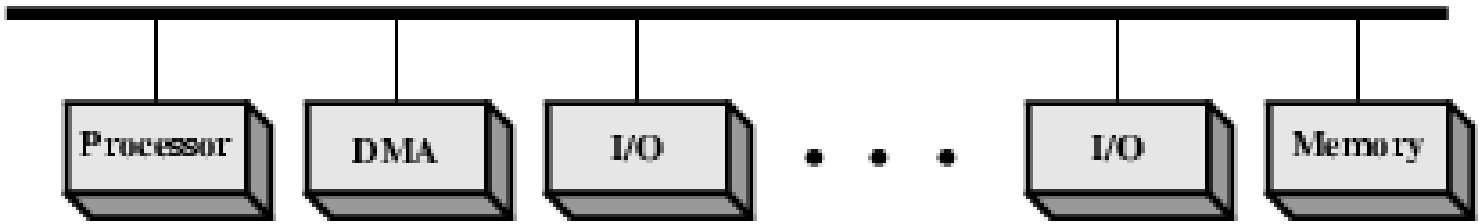
- Programa chequeando estado de cada controlador.

Determinar el manejador a ejecutar

- Identificación por software.
- Identificación por hardware.
  - Vector de direcciones (8086).
  - Vector de instrucciones (Sparc).

Introducción

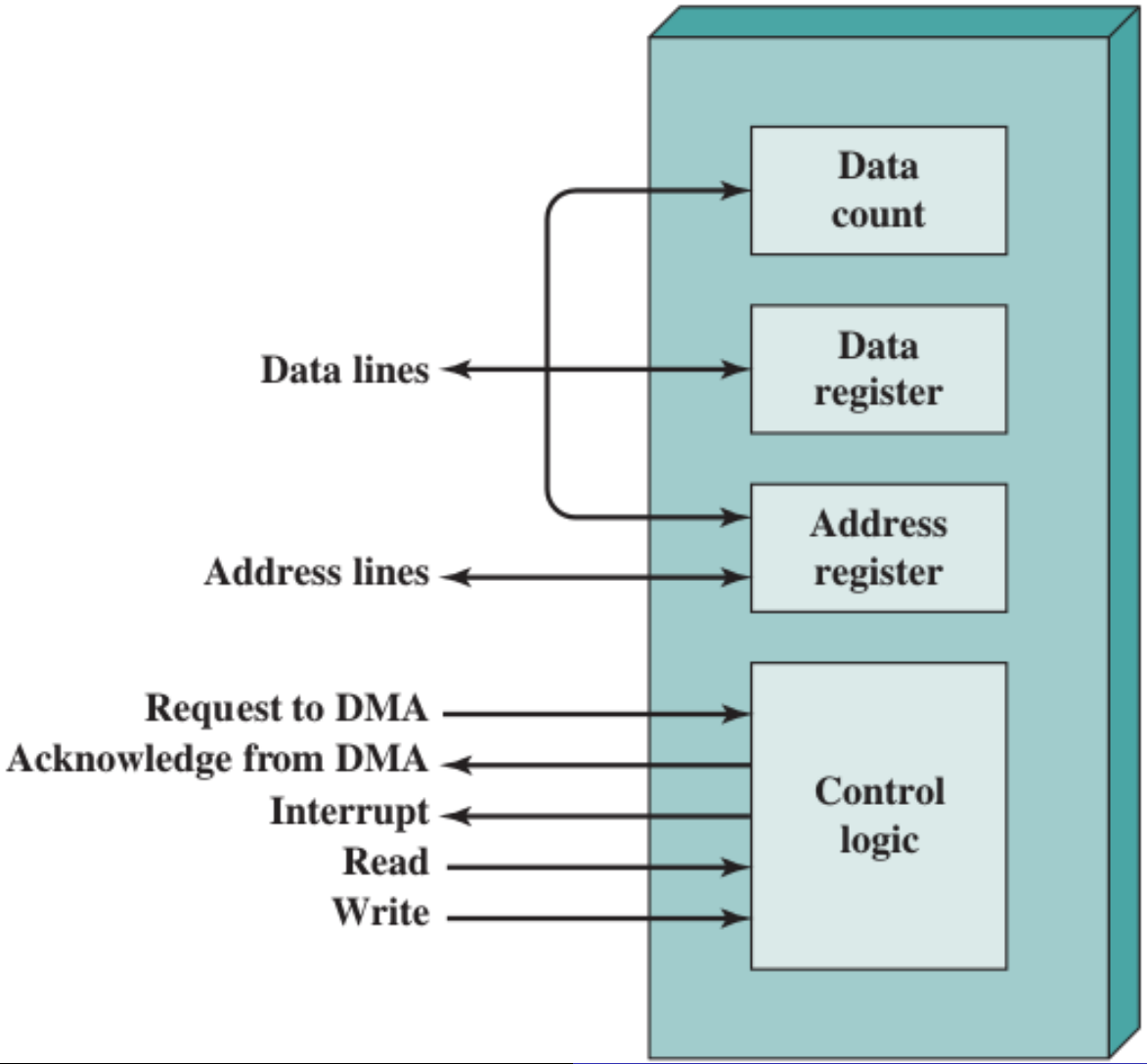
- E/S programada y por interrupciones requieren la intervención activa de la CPU
  - Limita la velocidad de transferencia
  - CPU "queda atada" a la E/S
- Alternativa DMA
  - Módulo adicional (hardware) en el bus
  - El controlador de DMA "sustituye" a la CPU en tareas de E/S



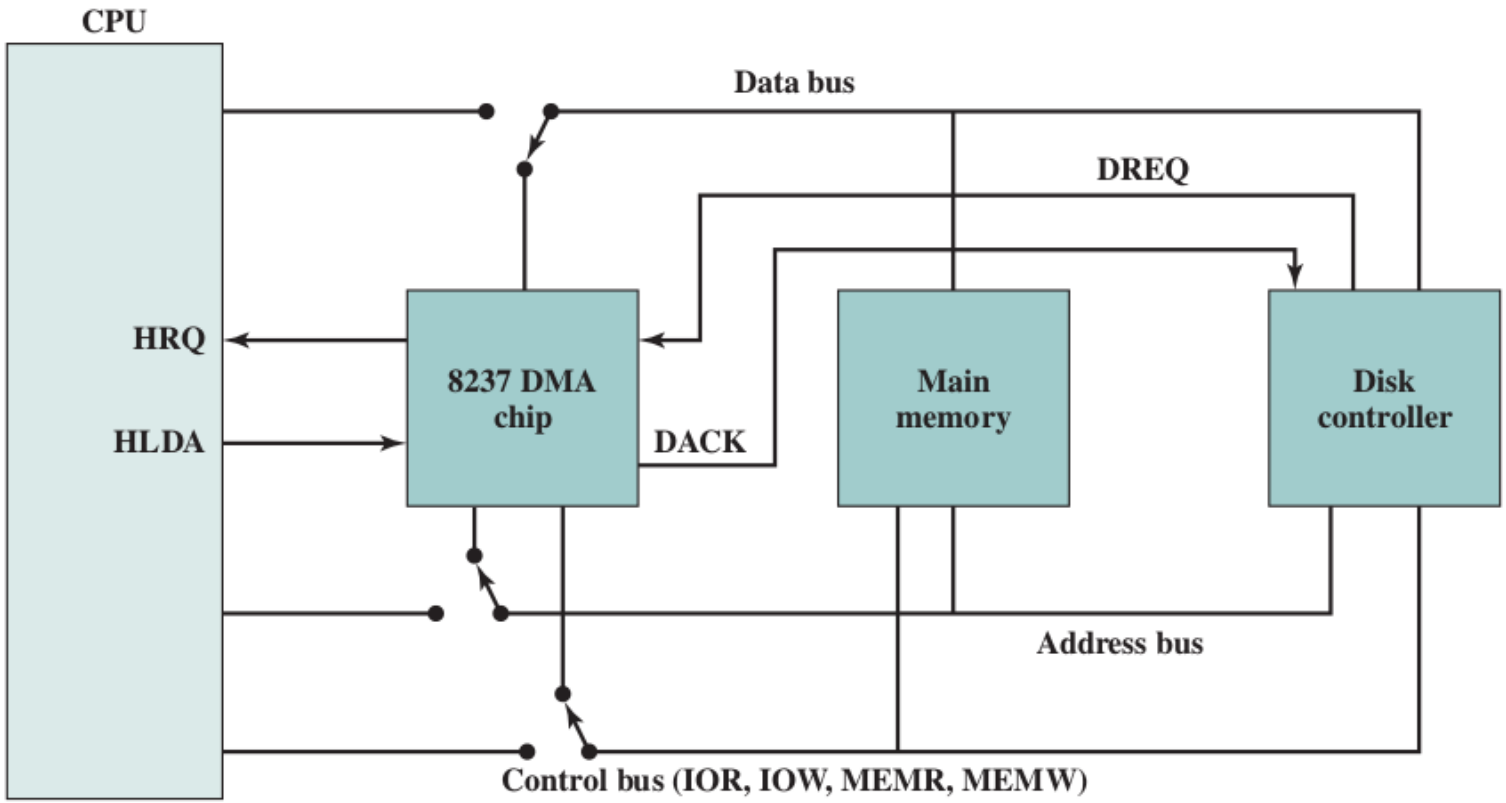
Cómo funciona el DMA

- La CPU programa el controlador de DMA:
  - Lectura/Escritura
  - Dirección del dispositivo
  - Dirección de comienzo del bloque de memoria
  - Cantidad de datos a transferir
- La CPU hace otra tarea mientras el controlador de DMA se encarga de la transferencia
- El controlador de DMA interrumpe cuando finaliza
- "Robo" de ciclos del DMA
  - Por cada palabra transferida, el DMA se adueña del bus
  - No es una interrupción (no hay que cambiar de contexto)
  - La CPU queda suspendida si necesita acceder al bus
  - Puede enlentecer la CPU, pero no tanto como si la CPU tuviera que hacer la transferencia

Direct Memory Access

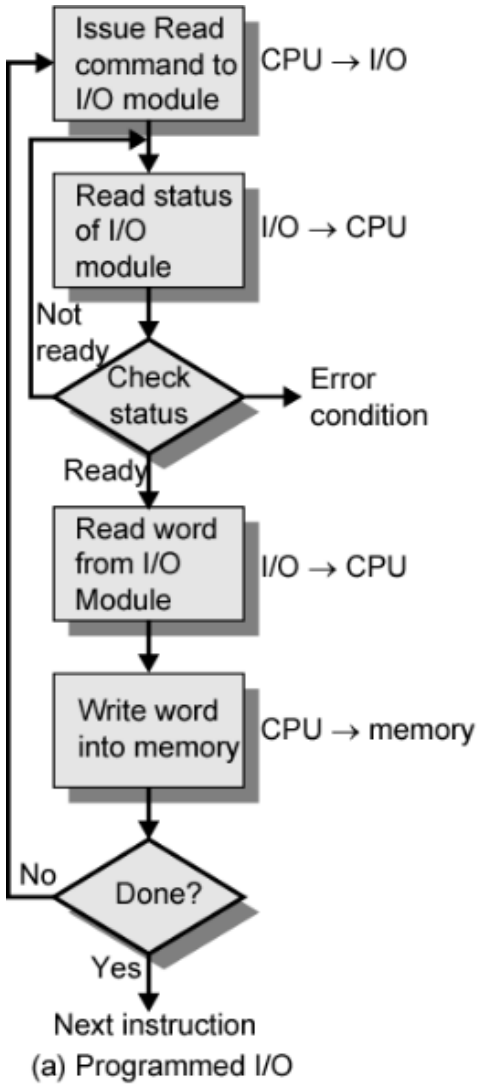


Direct Memory Access

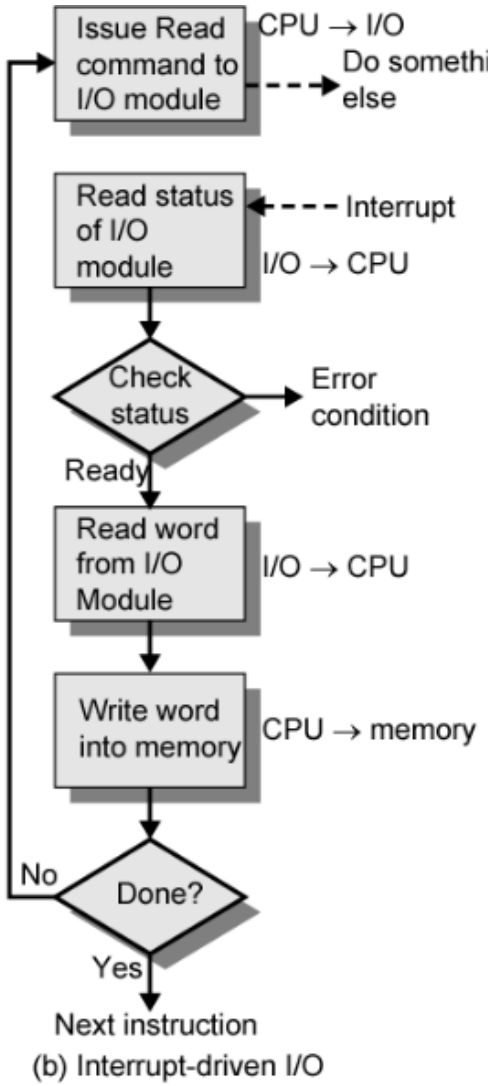


DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request

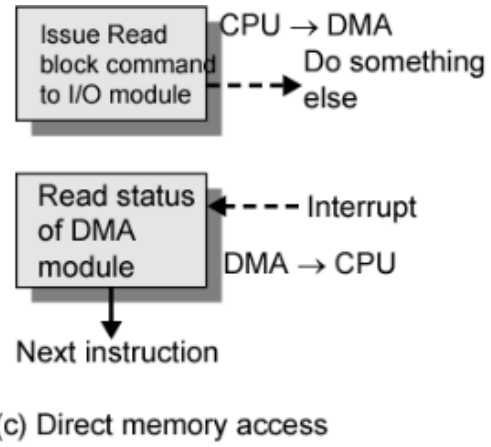
Técnicas para leer un bloque de datos



(a) Programmed I/O



(b) Interrupt-driven I/O



(c) Direct memory access

Lecturas Recomendadas

Departamento de arquitectura.

Interrupciones.

Notas de teórico. 2011.

Preguntas