Introducción al 8086
Parte II

Facultad de Ingeniería

Universidad de la República

Instituto de Computación

Rutinas recursivas(1/10)

- Introducción
- Salvar contexto.
- Manejo adecuado del stack.
- Atención con la dirección de retorno.
- Limitadas por el tamaño del stack.

Rutinas recursivas(2/10)

- Ejemplo: Función Factorial
- Definición

```
factorial(0)=1
factorial(n)= factorial(n-1)*n
```

- Alto Nivel

```
function factorial(n: short):short;
  begin
   if (n=0) then factorial:=1
   else factorial:= factorial(n-1)*n;
  end
```

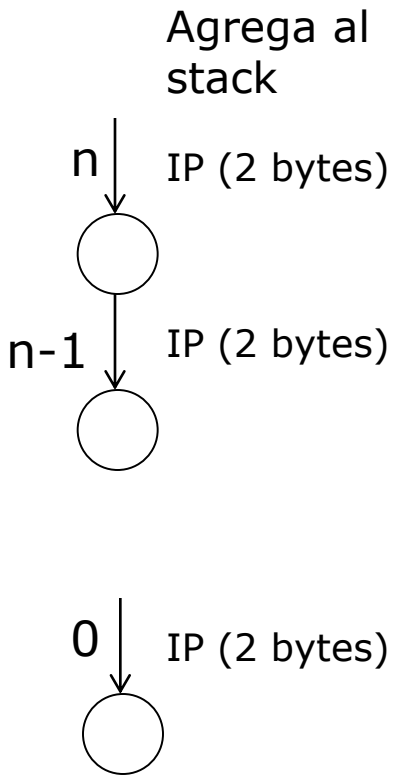
Rutinas recursivas(3/10)

Llamada	Asembler

mov ax,n
call fact
mov resultado,bx
	fact proc
	  cmp ax,0 ; comparo n con cero
	  je esCero
	  dec ax ; ajusto parámetro para la invocación
	  call fact ; realizo la llamada recursiva
	  inc ax
	  mov cx,ax ; guardo ax pues mul lo modifica
	  mul bx ; calculo el paso recursivo
	  mov bx,ax ; asigno el resultado del paso
	  ; recursivo
	  mov ax,cx ; restauro ax
	  jmp fin
	esCero:
	  mov bx,1 ; asigno el resultado del paso base
	fin:
	  ret
	fact endp

Rutinas recursivas(4/10)

- Cálculo del consumo de stack
- Planteo de la recurrencia
  - consumo(0)=2
  - consumo(n)=2+consumo(n-1)
- Resolviendo la recurrencia
  - consumo(n)=2*(n+1)



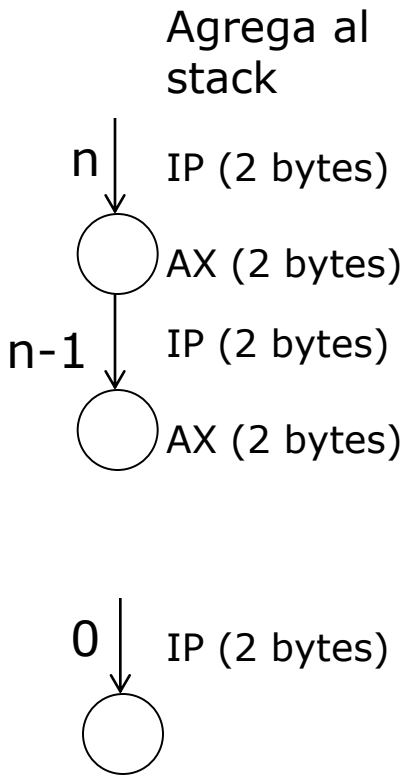
Rutinas recursivas(5/10)

Llamada	Asembler

mov ax,n
call fact
mov resultado,bx
	fact proc
	  cmp ax,0 ; comparo n con cero
	  je esCero
	  push ax ; salvo contexto
	  dec ax ; ajusto parámetro para la invocación
	  call fact ; realizo la llamada recursiva
	  pop ax ; restauro el contexto
	  mul bx ; calculo el paso recursivo
	  mov bx,ax ; asigno el resultado del paso
	  ; recursivo
	  jmp fin
	esCero:
	  mov bx,1 ; asigno el resultado del paso base
	fin:
	  ret
	fact endp

Rutinas recursivas(6/10)

- Cálculo del consumo de stack
- Planteo de la recurrencia
  - consumo(0)=2
  - consumo(n)=4+consumo(n-1)
- Resolviendo la recurrencia
  - consumo(n)=4*n+2



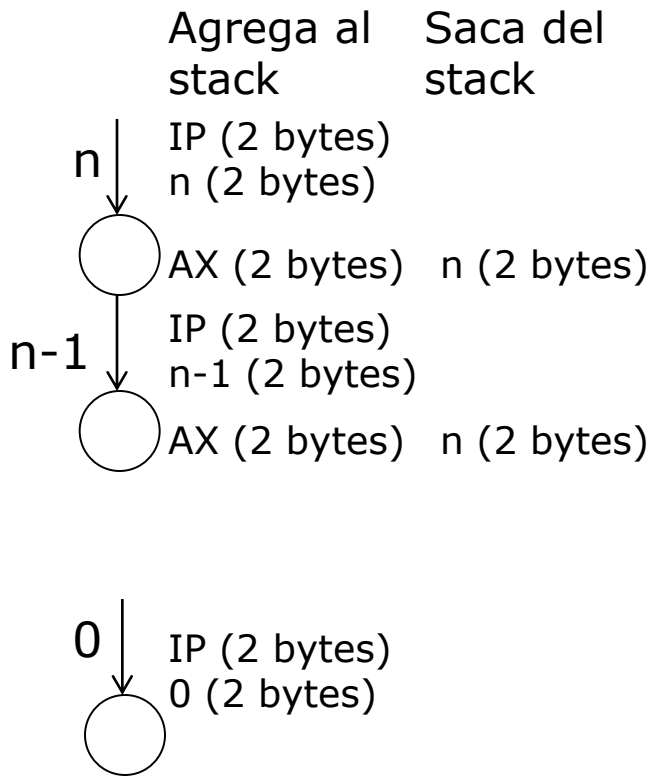
Rutinas recursivas(7/10)

Llamada	Asembler

push n
call fact
pop resultado
	fact proc
	  pop dx ; saco dirección de retorno del stack
	  pop ax ; saco parámetro del stack
	  push dx ; coloco la dirección de retorno en el stack
	  cmp ax,0 ; comparo n con cero
	  je esCero
	  push ax ; guardo el contexto
	  dec ax ; ajusto parámetro para la invocación
	  push ax ; coloco parámetro en el stack
	  call fact ; realizo la llamada recursiva
	  pop bx ; retiro del stack el resultado de fact(n-1)
	  pop ax ; restauro el contexto
	  mul bx ; calculo el paso recursivo
	  mov bx,ax ; asigno el resultado del paso recursivo
	  jmp fin
	esCero:
	  mov bx,1 ; asigno el resultado del paso base
	fin: ; acomodo el stack para el retorno
	  pop dx ; saco dirección de retorno del stack
	  push bx ; coloco el resultado en el stack
	  push dx ; coloco la dirección de retorno en el stack
	  ret
	fact endp

Rutinas recursivas(8/10)

- Cálculo del consumo de stack
- Planteo de la recurrencia
  - consumo(0)=4
  - consumo(n)=4+consumo(n-1)
- Resolviendo la recurrencia
  - consumo(n)=4*n+4



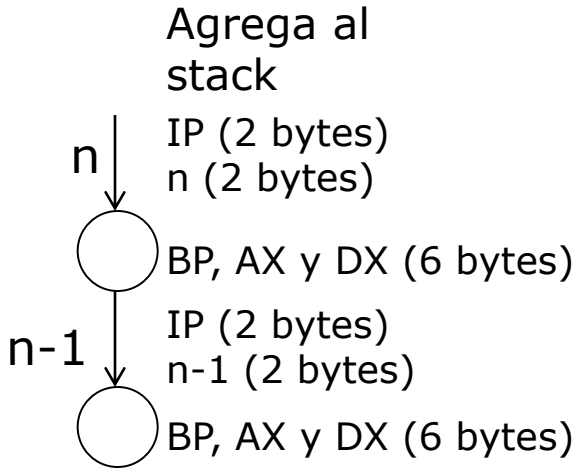
Rutinas recursivas(9/10)

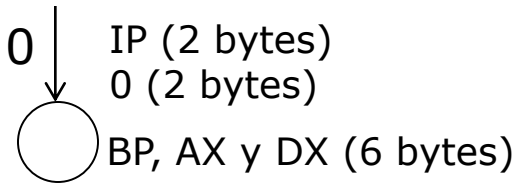
Llamada	Asembler

push n
call fact
pop resultado
	fact proc
	  push bp ; guardo el registro bp
	  mov bp,sp ; apunto con bp al tope de la pila
	  push ax ; conservo registros
	  push dx
	  cmp word ptr [bp+4],0 ; comparo n con cero
	  je esCero
	  mov ax,[bp+4] ; obtengo n
	  dec ax ; ajusto parámetro para la invocación
	  push ax ; coloco parámetro en el stack
	  call fact ; realizo la llamada recursiva
	  pop ax ; obtengo el resultado
	  mul word ptr [bp+4] ; calculo el paso recursivo
	  jmp fin
	esCero:
	  mov ax,1 ; asigno el resultado del paso base
	fin: ; acomodo el stack para el retorno
	  ;piso el parámetro de entrada con el resultado
	  mov [bp+4],ax
	  pop dx ; restauro registros
	  pop ax
	  pop bp
	  ret
	fact endp

Rutinas recursivas(10/10)

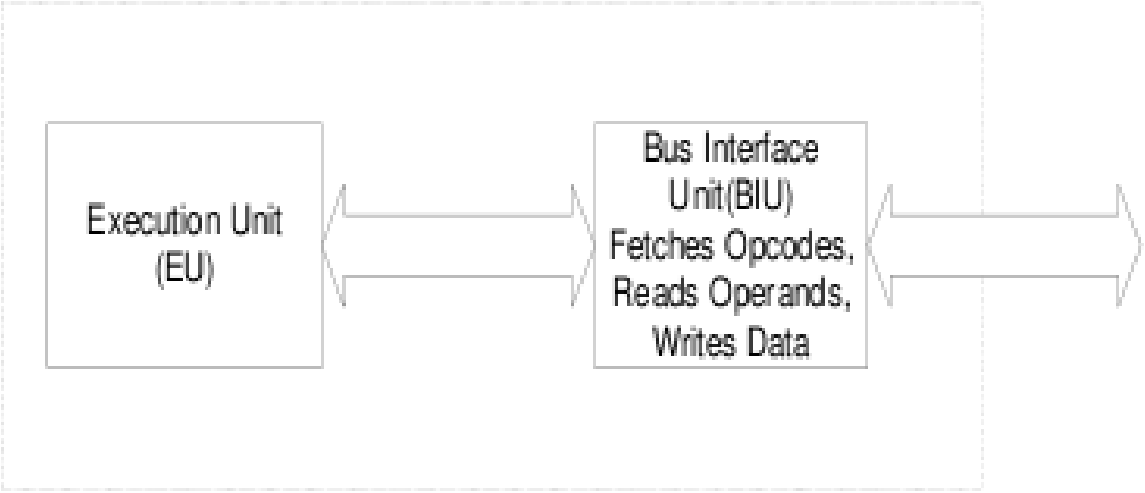
- Cálculo del consumo de stack
- Planteo de la recurrencia
  - consumo(0)=10
  - consumo(n)=10+consumo(n-1)
- Resolviendo la recurrencia
  - consumo(n)=10*(n+1)





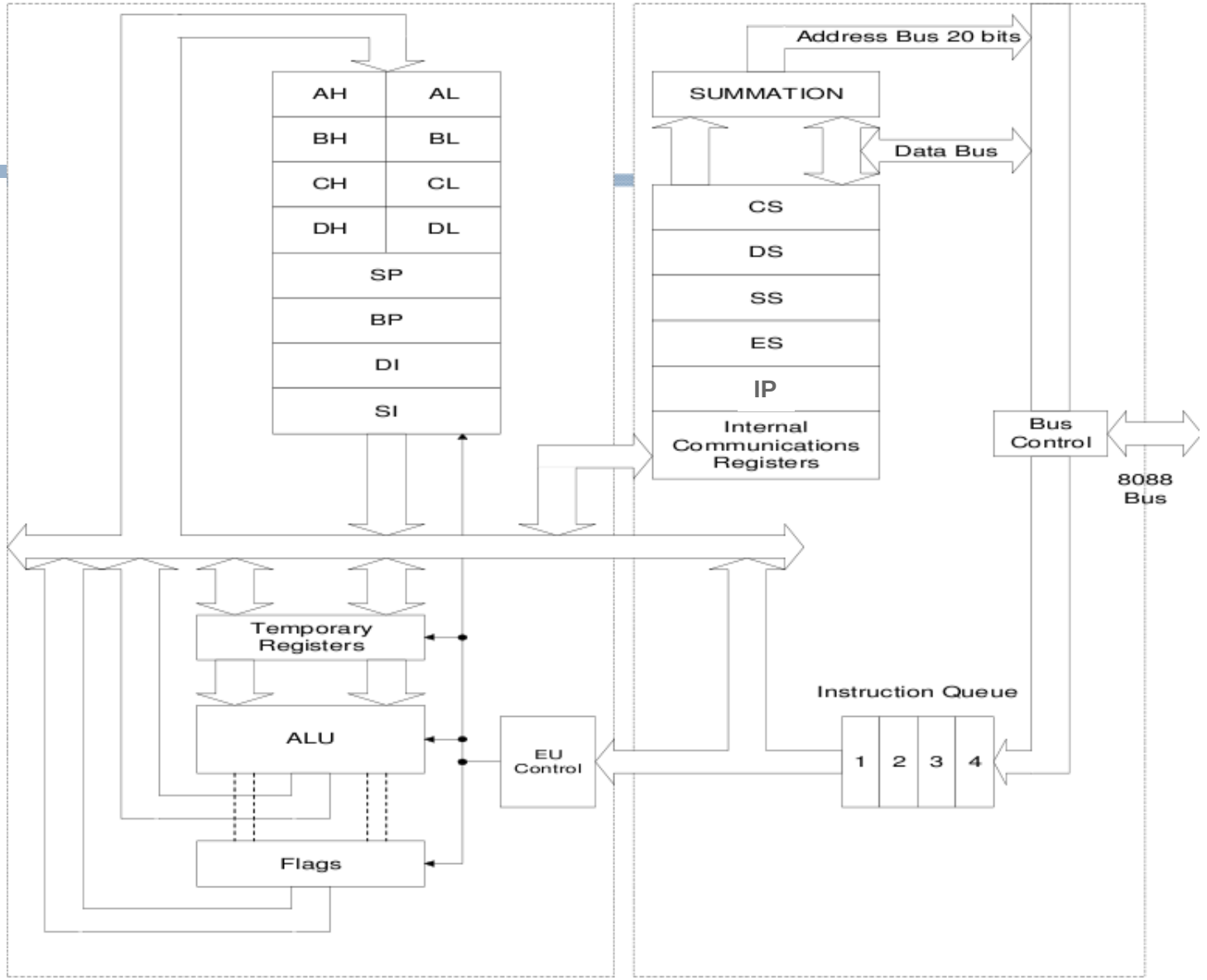
Unidades funcionales del 8086/8088

- Execution unit (EU) – ejecuta las instrucciones
- Bus interface unit (BIU) – fetch de instructiones y operandos, escritura de resultados
- Prefetch queue: 8086/6 bytes, 8088/4 bytes



8086/8088 MPU

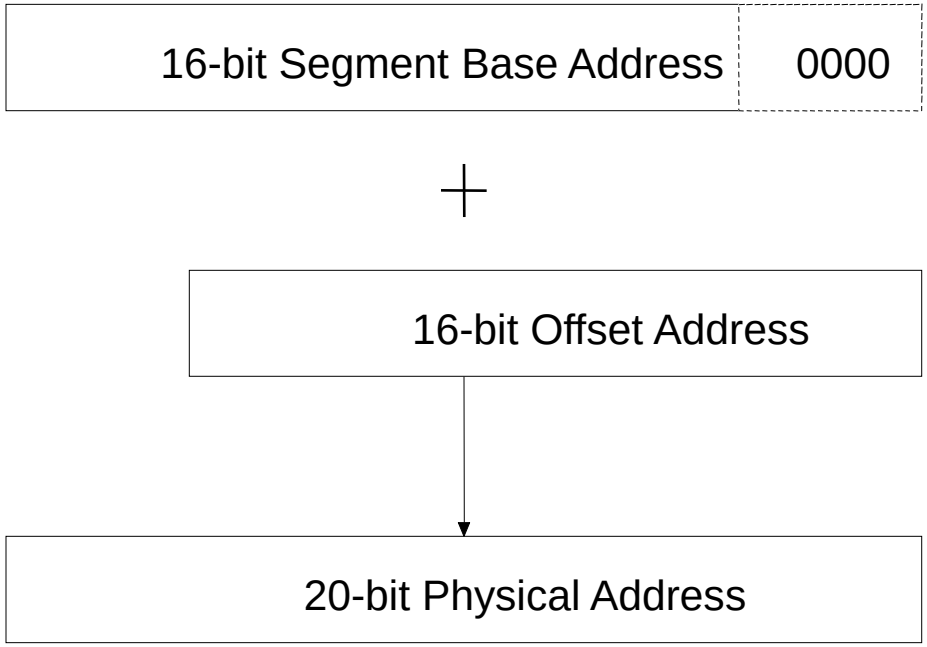
Organización Interna del 8086/8088

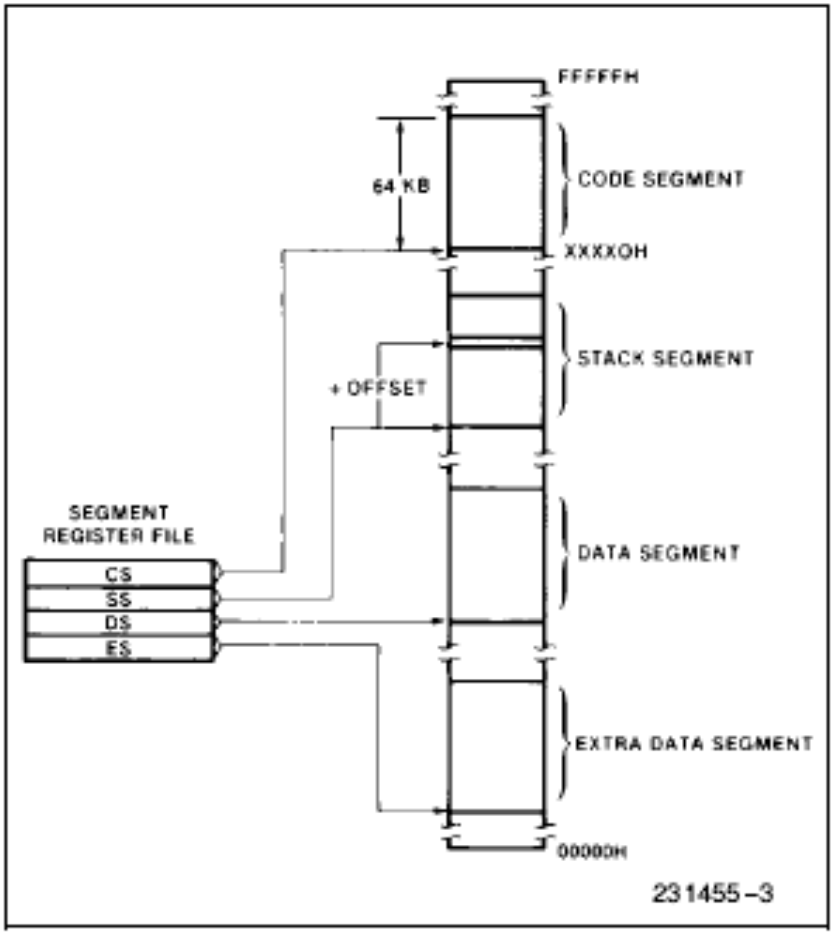


Elementos de la BIU

- Instruction Queue: la próxima instrucción u operando puede ser leído desde memoria mientras el procesador ejecuta la instrucción corriente
- Dado que la interfaz de memoria es mucho más lenta que el procesador, la cola de instrucciones mejora la performance global del sistema.
- Registros de Segmento:
- CS, DS, SS y ES: registers de 16 bit
- Usados como base para generar las direcciones de 20 bits
- Permiten al 8086/8088 direccionar 1Mbyte de memoria
- El programa puede utilizarlos para apuntar a diferentes segmentos durante la ejecución
- Instruction Pointer (IP) contiene el offset de la dirección de la próxima instrucción (distancia en bytes desde la dirección contenida en el registro CS)

Direcciones de 20 bits en el 8086/8088

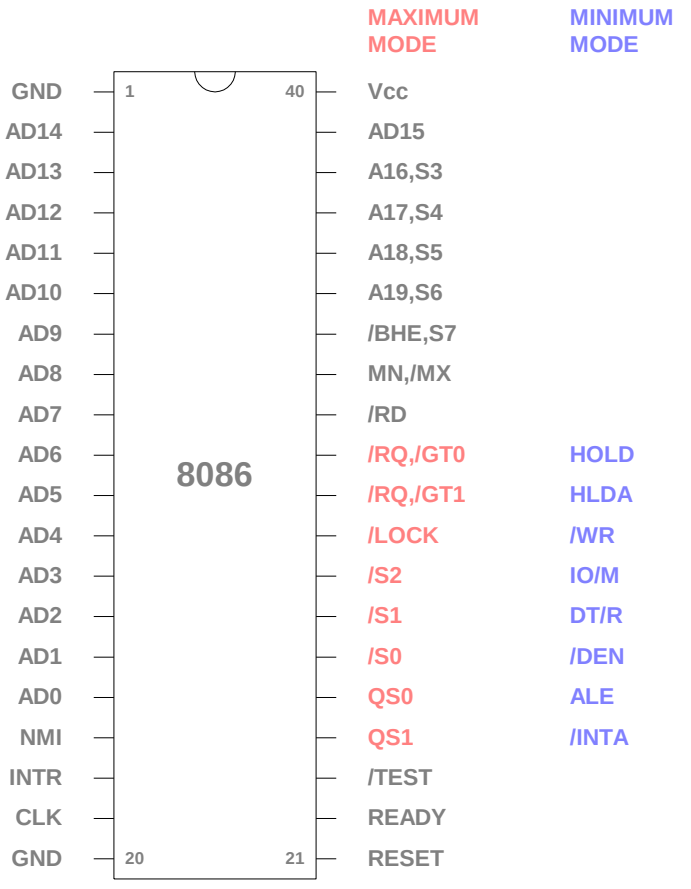


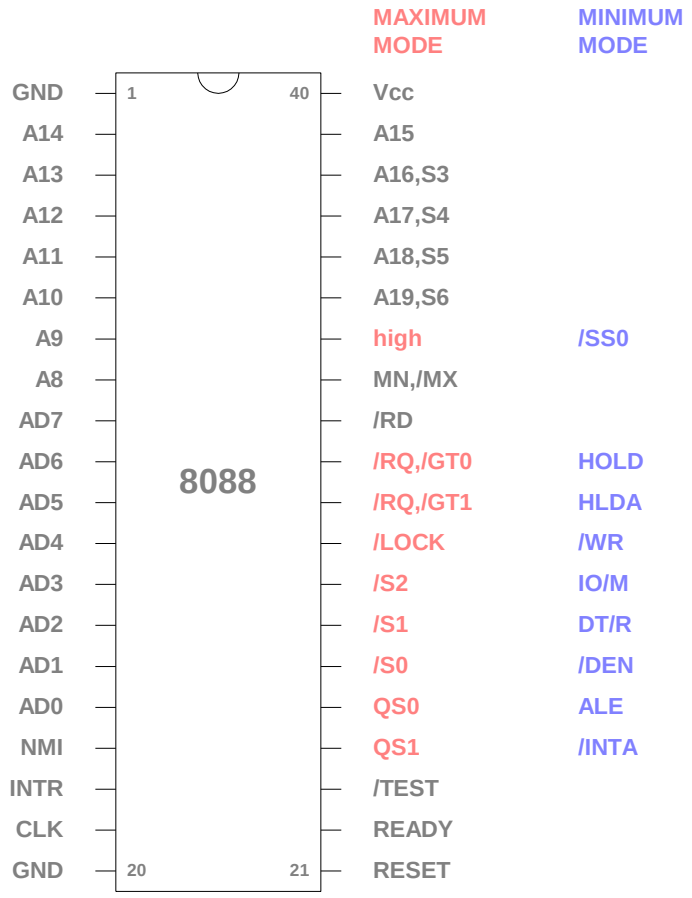


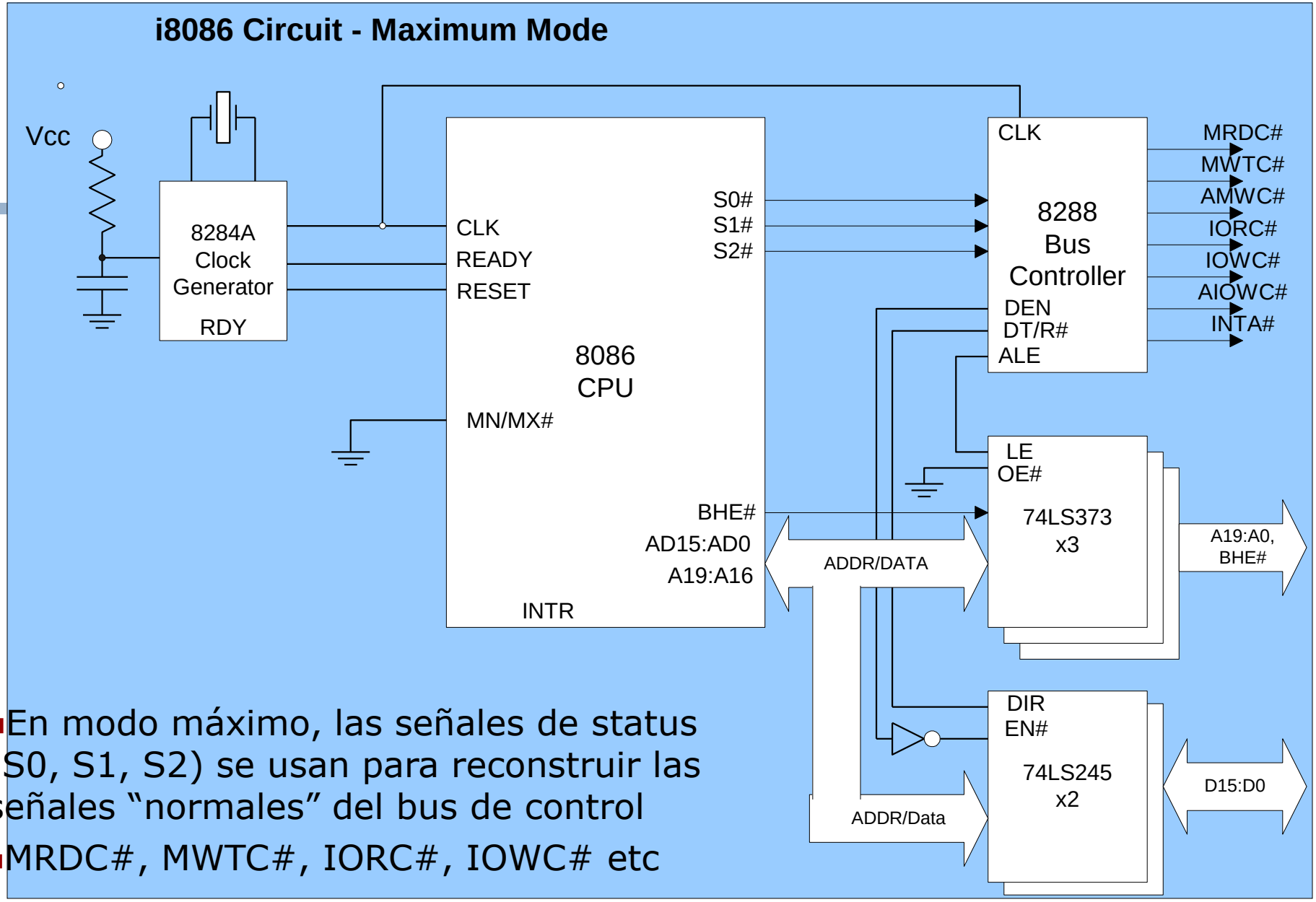
Construyendo un sistema basado en el 8086/8

- 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
  - Mínimo
- El Modo Máximo el 8086/8 necesita *al menos* los siguientes circuitos extra: 8288 Bus Controller, 8284A Clock Generator, 74HC373s y 74HC245s

Modos de funcionamiento







- En modo máximo, las señales de status (S0, S1, S2) se usan para reconstruir las señales "normales" del bus de control
- MRDC#, MWTC#, IORC#, IOWC# etc

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 0FFFF0h (CS=0F000h, IP=0FFF0h)
  - 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

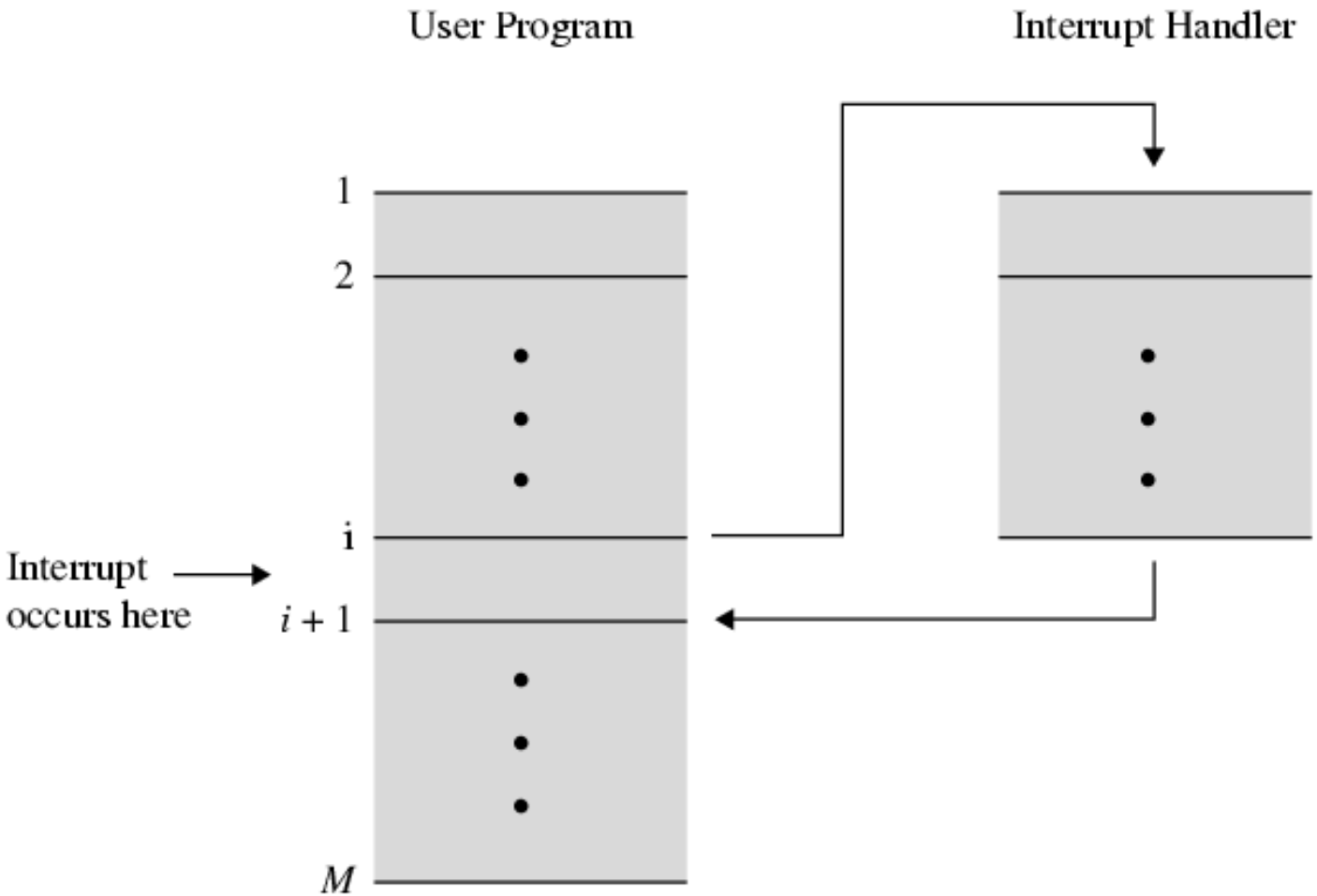
Interrupciones

- Mecanismo que permite interrumpir la secuencia normal de procesamiento de la CPU ante condiciones particulares
- Pueden ser de alguna de las siguientes clases:
- Programa
  - Ej. overflow, división por cero, protección de memoria
- Timer
  - Generado por un temporizador interno del procesador
  - Usado en pre-emptive multi-tasking
- E/S
  - Provocada por el controlador de E/S
- Fallo de Hardware
  - Ej. Error de paridad de memoria

Ciclo de Interrupciones

- Se agrega al ciclo de instrucción
- Procesador chequea por la interrupción
- Indicado por una señal de interrupción
- Si no hay interrupción, fetch próxima instrucción
- Si hay una interrupción pendiente:
- Suspender ejecución del programa corriente
- Salvar contexto
- Hacer que PC apunte a la dirección de inicio del manejador de la interrupción
- Procesar interrupción
- Restaurar contexto y continuar el programa interrumpido

Transferencia de Control vía Interrupciones

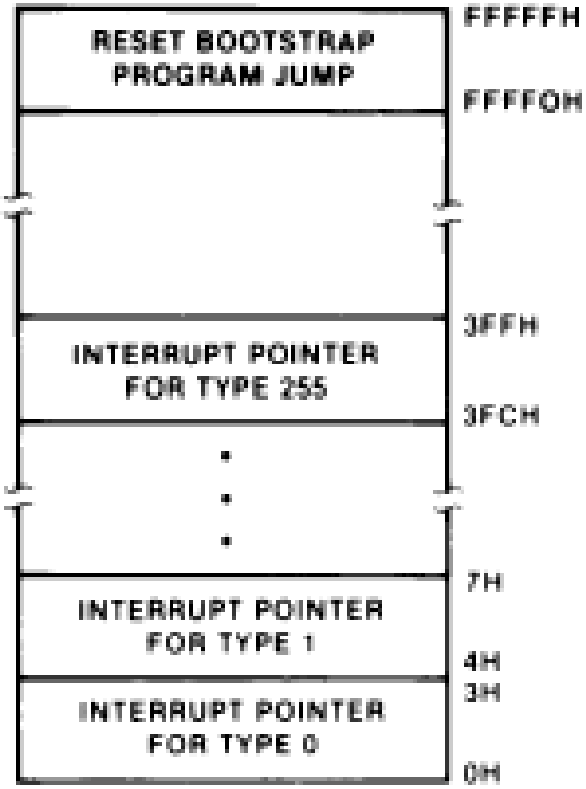


Interrupciones

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

Vector de Interrupciones

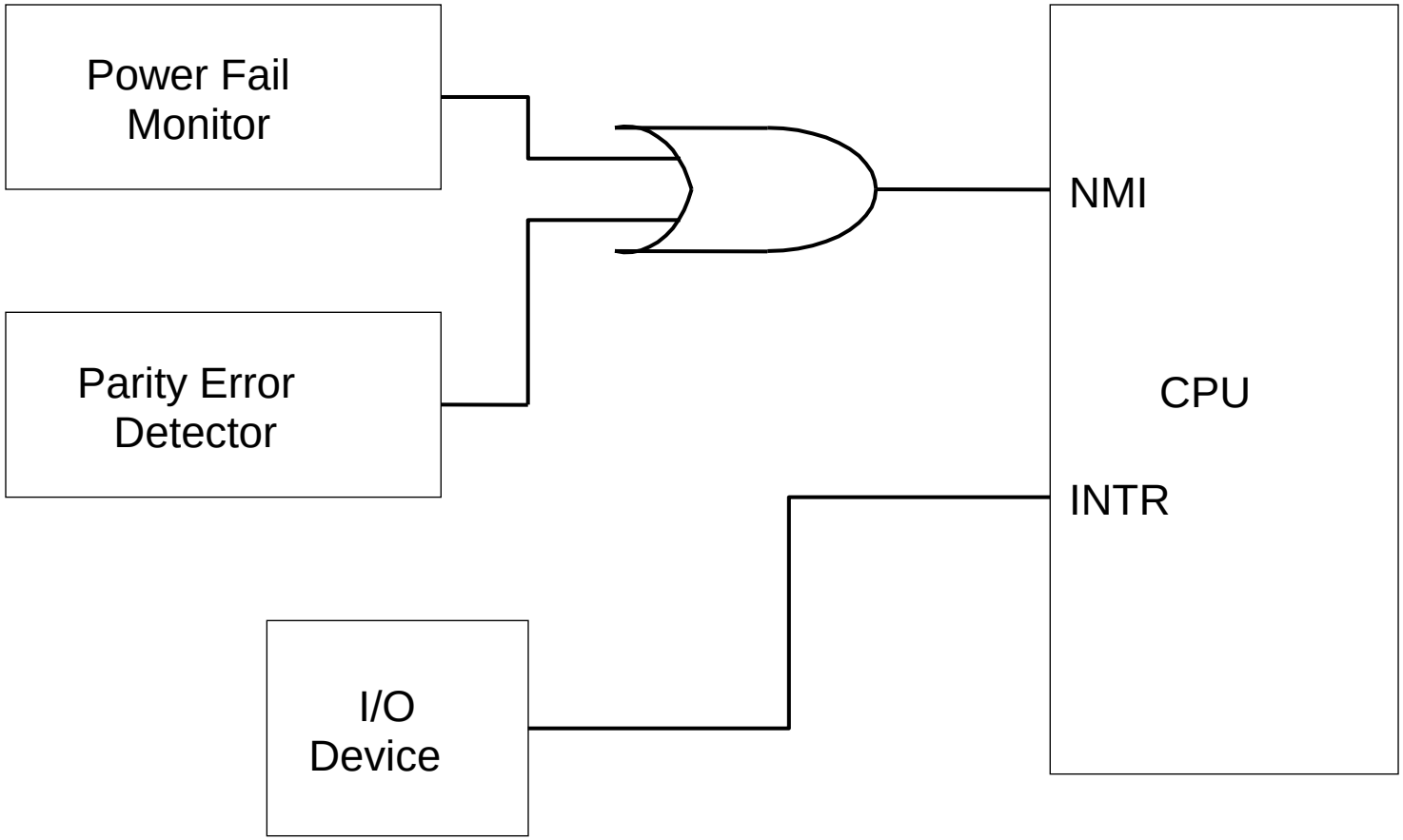
- Las localizaciones de memoria 00000H a 003FFH están reservadas para las interrupciones
- Existen 256 tipos posibles de interrupciones
- Cada Handler o Rutina de Servicio de Interrupción está direccionada por un puntero de 4 bytes: 16 bits de segmento y 16 bits de offset
- Las rutinas y los punteros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



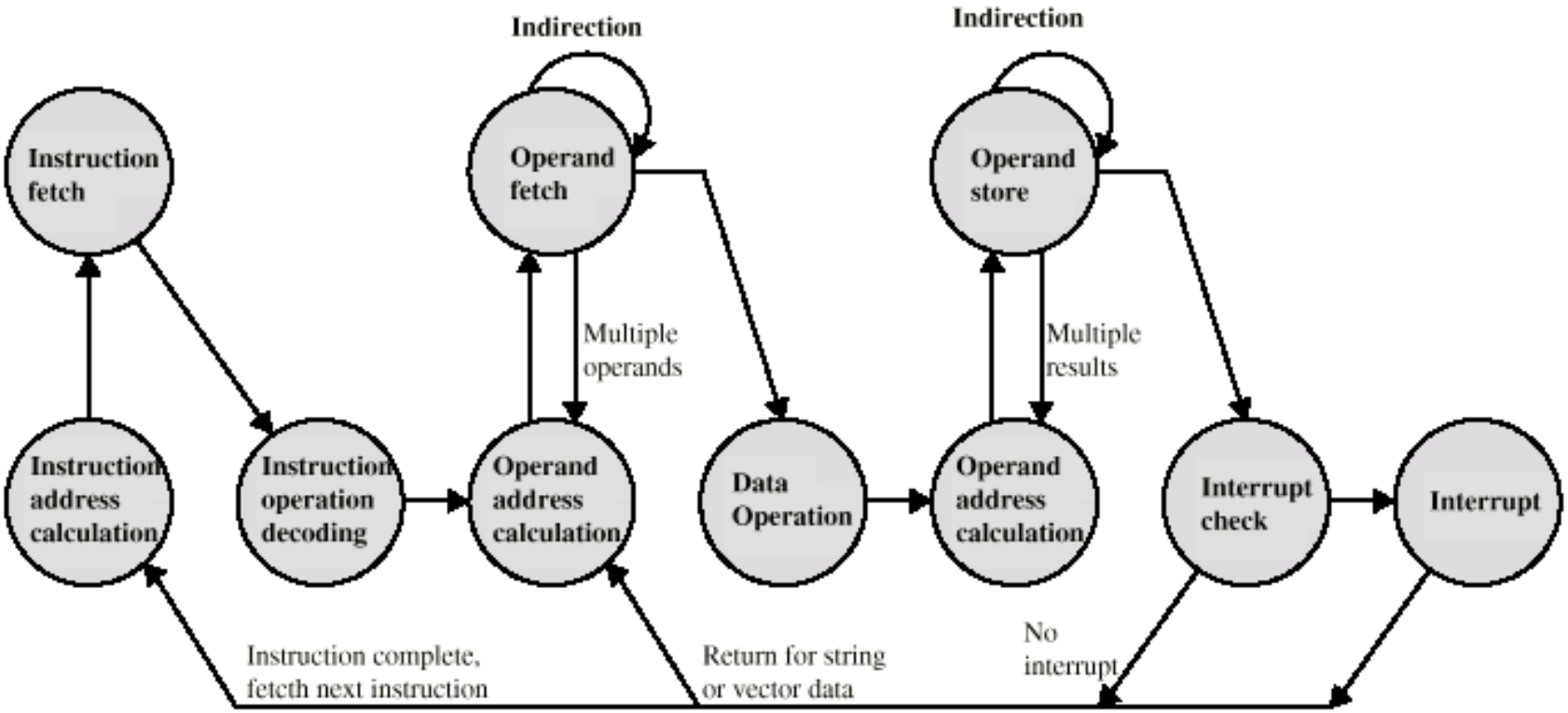
Interrupciones enmascarables y no enmascarables

- Las interrupciones pueden enmascararse globalmente usando la Interrupt Enable Flag (IE o I)
- IE 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…

Ejemplo de NMI



Ciclo de Instrucción

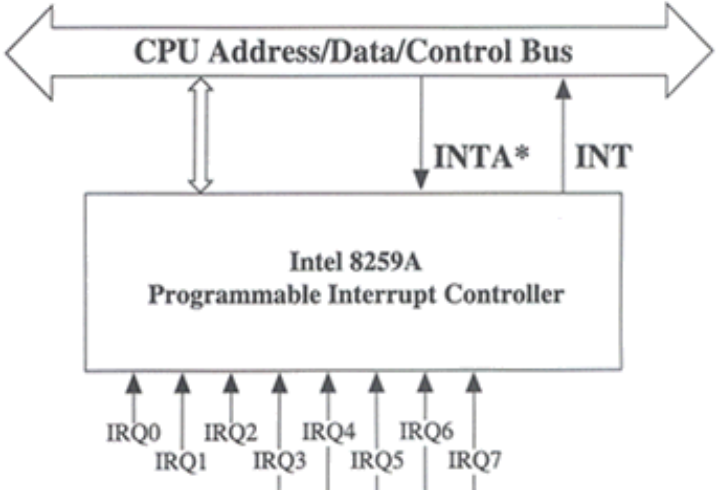


Ciclo de Interrupciones

- Se ubica al final del ciclo de instrucción
- Si IF el procesador chequea por la interrupción
- Indicado por una señal de interrupción
- Si no hay interrupción, fetch próxima instrucción
- Si hay una interrupción pendiente:
- Suspender ejecución del programa corriente
- Salvar contexto
- Hacer que PC apunte a la dirección de inicio del manejador de la interrupción
- Procesar interrupción
- Restaurar contexto y continuar el programa interrumpido

Controlador 8259

- Un dispositivo genera una señal de interrupción
- Si no hay otra interrupción de mayor prioridad ejecutandose 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



INT

Formato: INT op_1

Tipo Args: (i)

Lógica:

```
PUSHF
IF=0
TF=0
CALL FAR tabla_int( op1 )
```

Descripción: Genera una interrupción software tipo op_1.

Banderas:

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

IN

Formato:	IN	op_1,	op_2
Tipo Args:		(AL, AX)	(i,DX)

Lógica:

si $\mathrm{op}_1 = \mathbf{Al}$

$\mathbf{Al} = \mathrm{I/O}(\mathrm{op}_2)$

sino si $\mathrm{op}_1 = \mathbf{AX}$

$\mathbf{Ax} = \mathrm{I/O}(\mathrm{op}_2)$

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.

- OUT

CLC

Formato: CLC

Lógica: CF := 0

Descripción: Borra la bandera de acarreo (CF) sin afectar a ninguna otra bandera.

Banderas:

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

- Estas son las que importan: CLI y STI.

Interrupciones(1/5)

- Interrupciones de Hardware o Software
- 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
- Notas
- 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.

Interrupciones(2/5)

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

Interrupciones(3/5)

- Problema
- 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:
  1) El procesador no está dedicado a esta aplicación.
  2) La interrupción de TIMER es la 08h (elemento de índice 8 dentro del vector de interrupciones), y se ejecuta 18 veces por segundo.

Interrupciones(4/5)

- Alto nivel

```
var
  cantTics: byte;
  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
```

Interrupciones(5/5)

- Asembler

```
ES_LED equ …
INT8_OFF equ 8*4
INT8_SEG equ 8*4+2

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)
  cli         ; {deshabilitoInterrupciones}
  xor ax,ax
  xor bx,bx
  mov es,ax
  ;instaloManejadorTimer sin llamar al
  ;manejador actual
  ;indico desplazamiento y segmento
  mov word ptr es:[bx+INT8_OFF],offset TIEMPO
  mov  word ptr es:[bx+INT8_SEG],cs
  sti ; {habilitoInterrupciones}
principal endp
```

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

Referencias

- Notas del curso sobre 8086.