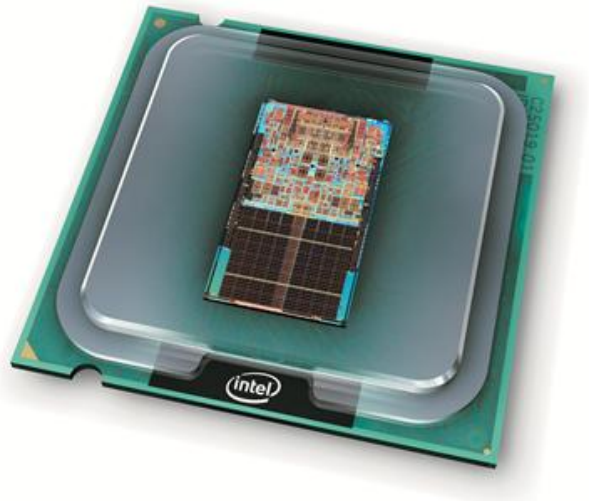


Procesadores IA-32 - Hardware Basis

Alejandro Furfaro

23 de junio de 2020

Temario

1. Arquitectura IA-32. Hardware Básico
   - Descripción funcional.
   - Organización Interna del 80386DX
   - Descripción de Terminales.

Temario

Procesador 80386DX

El 80386 es el primer miembro de 32 bits de la familia x86. No obstante el cambio de 16 a 32 bits en la Arquitectura completa del procesador, su código objeto es 100 % compatible con los procesadores anteriores de la familia, soportando transferencias de datos de 8 y 16 bits, pero agregando además la nueva capacidad de 32 bits que introduce esta nueva arquitectura.

Características

Características

- Soporte para tipos de datos de 32, 16 y 8 bits.

Características

- Soporte para tipos de datos de 32, 16 y 8 bits.
- Los 8 registros de propósito general se han ampliado a 32 bits, conservando la capacidad de trabajar como registros de 16 bits, tal como se los conocía en el 8086 y 80286, y además 4 de ellos pueden tratarse como pares de registros independientes de 8 bits.

Características

- Soporte para tipos de datos de 32, 16 y 8 bits.
- Los 8 registros de propósito general se han ampliado a 32 bits, conservando la capacidad de trabajar como registros de 16 bits, tal como se los conocía en el 8086 y 80286, y además 4 de ellos pueden tratarse como pares de registros independientes de 8 bits.
- Se conserva en su totalidad el modelo de programación de 16 bits pre existente.

Características

- Soporte para tipos de datos de 32, 16 y 8 bits.
- Los 8 registros de propósito general se han ampliado a 32 bits, conservando la capacidad de trabajar como registros de 16 bits, tal como se los conocía en el 8086 y 80286, y además 4 de ellos pueden tratarse como pares de registros independientes de 8 bits.
- Se conserva en su totalidad el modelo de programación de 16 bits pre existente.
- Bus de datos de 32 líneas. Realiza transacciones de este ancho en un solo ciclo de lectura, ganando en capacidad de transferencia respecto de sus predecesores 80286, 80886, etc.

Características

- Soporte para tipos de datos de 32, 16 y 8 bits.
- Los 8 registros de propósito general se han ampliado a 32 bits, conservando la capacidad de trabajar como registros de 16 bits, tal como se los conocía en el 8086 y 80286, y además 4 de ellos pueden tratarse como pares de registros independientes de 8 bits.
- Se conserva en su totalidad el modelo de programación de 16 bits pre existente.
- Bus de datos de 32 líneas. Realiza transacciones de este ancho en un solo ciclo de lectura, ganando en capacidad de transferencia respecto de sus predecesores 80286, 80886, etc.
- Bus de Address de 32 líneas que le permite direccionar 2^{32} direcciones de memoria de 1 byte, es decir 4 Gbytes de memoria física (su antecesor, el 80286, solo direccionaba 16 Mbytes de memoria física a través de sus 24 líneas de address, y 1 Gbyte en el modo protegido).

Características

Características

- En el modo protegido el 386 puede gestionar 64 Tbytes de memoria virtual.

Características

- En el modo protegido el 386 puede gestionar 64 Tbytes de memoria virtual.
- Las líneas de Address son independientes de las de datos, por lo tanto, al igual que el 80286, ya no requiere la señal **ALE**.

Características

- En el modo protegido el 386 puede gestionar 64 Tbytes de memoria virtual.
- Las líneas de Address son independientes de las de datos, por lo tanto, al igual que el 80286, ya no requiere la señal **`ALE`**.
- Unidad de gestión de memoria para el modo protegido que extiende las capacidades a 32 bits, y agrega al mecanismo de segmentación, una Unidad de Paginación que le permite implementar memory mapping de acuerdo con loa Sistemas Operativos Multitasking modernos, implementando sistemas de memoria virtual de manera mucho mas eficiente.

Características

- En el modo protegido el 386 puede gestionar 64 Tbytes de memoria virtual.
- Las líneas de Address son independientes de las de datos, por lo tanto, al igual que el 80286, ya no requiere la señal **ALE**.
- Unidad de gestión de memoria para el modo protegido que extiende las capacidades a 32 bits, y agrega al mecanismo de segmentación, una Unidad de Paginación que le permite implementar memory mapping de acuerdo con loa Sistemas Operativos Multitasking modernos, implementando sistemas de memoria virtual de manera mucho mas eficiente.
- Pipeline mejorado, aumentando las sub unidades que trabajan en paralelo, lo que le permite reducir el tiempo de ejecución de todas las instrucciones respecto de sus predecesores.

Características

Características

- Hardware de soporte a debugging (Debug Registers que permiten breakpoint programables, y por lo tanto posibilitan establecer un breakpoint en ROM si fuese necesario.

Características

- Hardware de soporte a debugging (Debug Registers que permiten breakpoint programables, y por lo tanto posibilitan establecer un breakpoint en ROM si fuese necesario.
- Modo Virtual 8086, que le permite correr aplicaciones desarrolladas para 8086/8088, es decir aplicaciones de modo real en Modo Protegido sin ningún tipo de cambio.

Características

- Hardware de soporte a debugging (Debug Registers que permiten breakpoint programables, y por lo tanto posibilitan establecer un breakpoint en ROM si fuese necesario.
- Modo Virtual 8086, que le permite correr aplicaciones desarrolladas para 8086/8088, es decir aplicaciones de modo real en Modo Protegido sin ningún tipo de cambio.
- Clock de 20, 25, y 33 Mhz.

Características

- Hardware de soporte a debugging (Debug Registers que permiten breakpoint programables, y por lo tanto posibilitan establecer un breakpoint en ROM si fuese necesario.
- Modo Virtual 8086, que le permite correr aplicaciones desarrolladas para 8086/8088, es decir aplicaciones de modo real en Modo Protegido sin ningún tipo de cambio.
- Clock de 20, 25, y 33 Mhz.
- Incorporación de Memoria Cache para optimizar performance.

Características

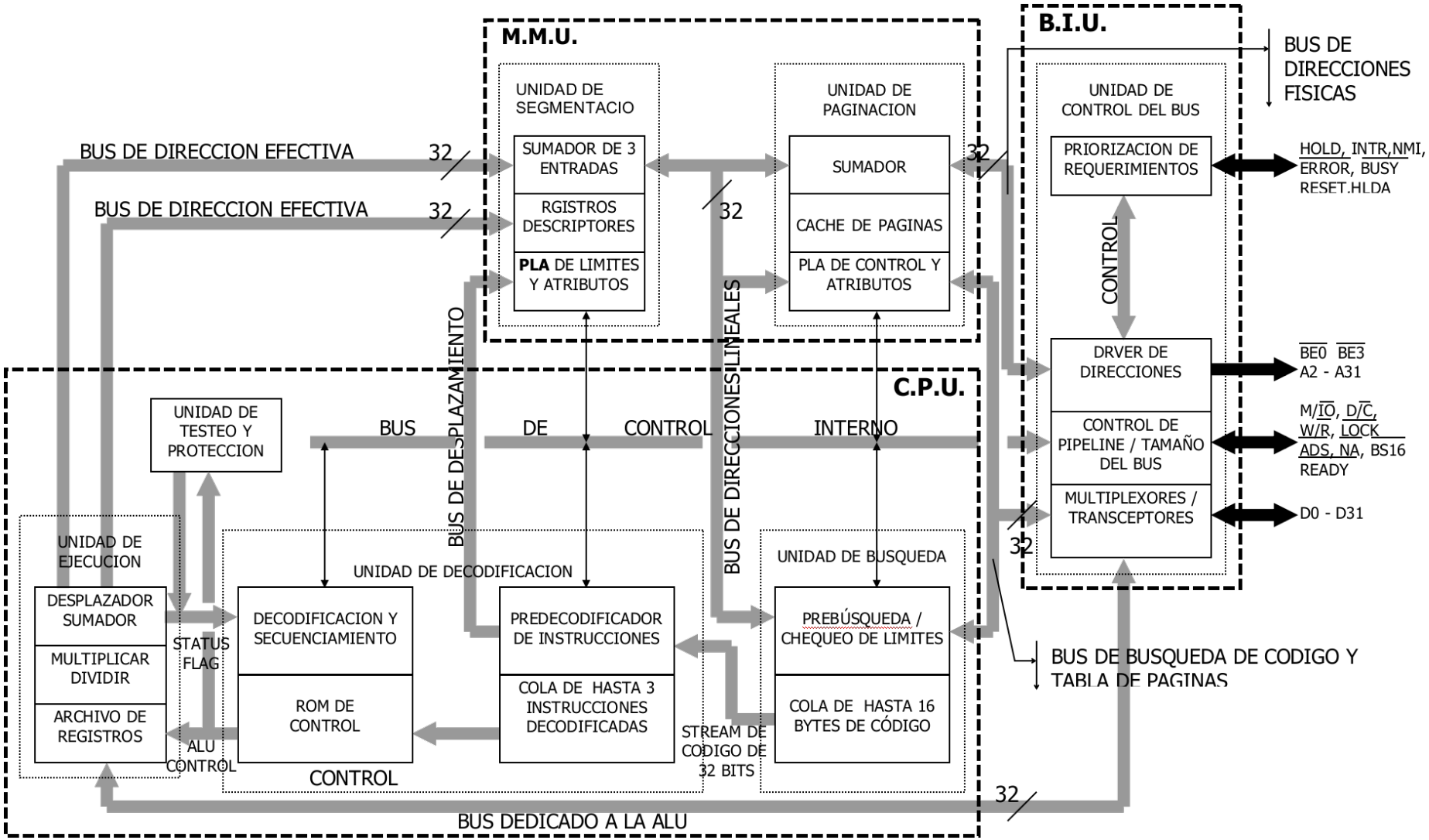
- Hardware de soporte a debugging (Debug Registers que permiten breakpoint programables, y por lo tanto posibilitan establecer un breakpoint en ROM si fuese necesario.
- Modo Virtual 8086, que le permite correr aplicaciones desarrolladas para 8086/8088, es decir aplicaciones de modo real en Modo Protegido sin ningún tipo de cambio.
- Clock de 20, 25, y 33 Mhz.
- Incorporación de Memoria Cache para optimizar performance.
- Coprocesador 80387, para cálculo en punto flotante.

Características

- Hardware de soporte a debugging (Debug Registers que permiten breakpoint programables, y por lo tanto posibilitan establecer un breakpoint en ROM si fuese necesario.
- Modo Virtual 8086, que le permite correr aplicaciones desarrolladas para 8086/8088, es decir aplicaciones de modo real en Modo Protegido sin ningún tipo de cambio.
- Clock de 20, 25, y 33 Mhz.
- Incorporación de Memoria Cache para optimizar performance.
- Coprocesador 80387, para cálculo en punto flotante.
- Encapsulado PGA (Pin Grid Array) de 132 pines.

Temario

Diagrama interno



Tres grandes bloques- Seis sub unidades

CPU Central Process Unit

- Unidad de prebúsqueda.
- Unidad de decodificación.
- Unidad de Ejecución.

MMU Memory Management Unit

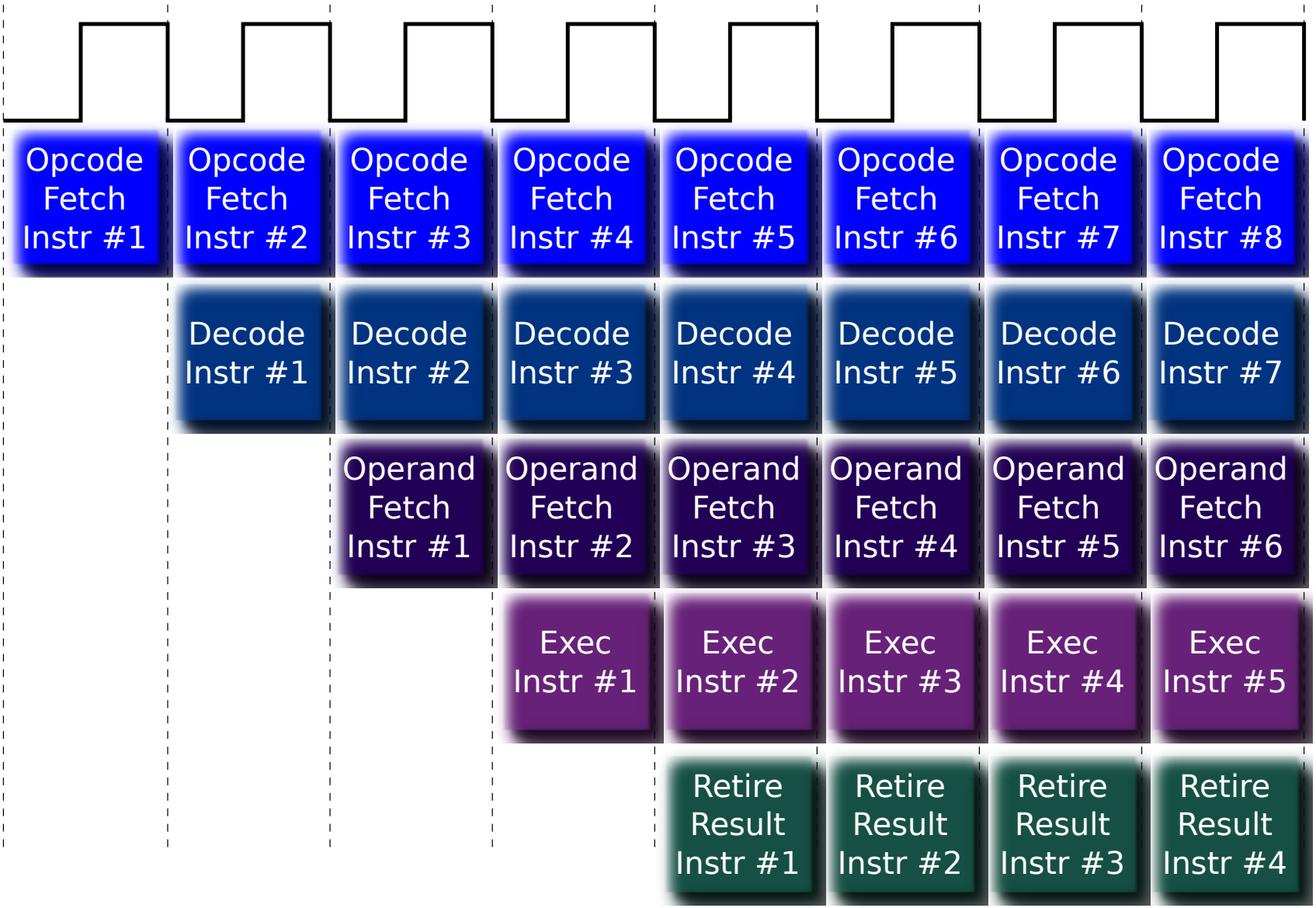
- Unidad de Segmentación.
- Unidad de Paginación.

BIU Bus Interface Unit

- Unidad de Control del Bus.

Los seis módulos enumerados trabajan en forma paralela, es decir, a pesar de trabajar independientemente lo hacen en forma coordinada. Esto es lo que denominamos ***pipeline***.

Pipeline


Opcode Fetch Instr #1
Opcode Fetch Instr #2
Opcode Fetch Instr #3
Opcode Fetch Instr #4
Opcode Fetch Instr #5
Opcode Fetch Instr #6
Opcode Fetch Instr #7
Opcode Fetch Instr #8
Decode Instr #1
Decode Instr #2
Decode Instr #3
Decode Instr #4
Decode Instr #5
Decode Instr #6
Decode Instr #7
Operand Fetch Instr #1
Operand Fetch Instr #2
Operand Fetch Instr #3
Operand Fetch Instr #4
Operand Fetch Instr #5
Operand Fetch Instr #6
Exec Instr #1
Exec Instr #2
Exec Instr #3
Exec Instr #4
Exec Instr #5
Retire Result Instr #1
Retire Result Instr #2
Retire Result Instr #3
Retire Result Instr #4

Unidad de Interfaz con el Bus

Unidad de Interfaz con el Bus

- Controla el intercambio de información entre el procesador y el exterior, a través de los buses.

Unidad de Interfaz con el Bus

- Controla el intercambio de información entre el procesador y el exterior, a través de los buses.
- Lee continuamente el código del programa a solicitud de la *Unidad de Pre búsqueda*, y los almacena en una cola de instrucciones **FIFO** (**F**irst **I**n **F**irst **O**ut) de 16 bytes.

Unidad de Interfaz con el Bus

- Controla el intercambio de información entre el procesador y el exterior, a través de los buses.
- Lee continuamente el código del programa a solicitud de la *Unidad de Pre búsqueda*, y los almacena en una cola de instrucciones **FIFO** (**F**irst **I**n **F**irst **O**ut) de 16 bytes.
- Busca en memoria los operandos para las instrucciones de modo de direccionamiento indirecto a solicitud de la *Unidad de Ejecución*.

Unidad de Interfaz con el Bus

- Controla el intercambio de información entre el procesador y el exterior, a través de los buses.
- Lee continuamente el código del programa a solicitud de la *Unidad de Pre búsqueda*, y los almacena en una cola de instrucciones **FIFO** (**F**irst **I**n **F**irst **O**ut) de 16 bytes.
- Busca en memoria los operandos para las instrucciones de modo de direccionamiento indirecto a solicitud de la *Unidad de Ejecución*.
- El bloque de Priorización de Requerimientos de la **BIU**, decide ante requerimientos simultáneos cual será cursado antes.

Unidad de Interfaz con el Bus

- Controla el intercambio de información entre el procesador y el exterior, a través de los buses.
- Lee continuamente el código del programa a solicitud de la *Unidad de Pre búsqueda*, y los almacena en una cola de instrucciones **FIFO** (**F**irst **I**n **F**irst **O**ut) de 16 bytes.
- Busca en memoria los operandos para las instrucciones de modo de direccionamiento indirecto a solicitud de la *Unidad de Ejecución*.
- El bloque de Priorización de Requerimientos de la **BIU**, decide ante requerimientos simultáneos cual será cursado antes.
- La búsqueda de Operandos tendrá la de más alta prioridad, y pospondrá cualquier otro acceso a memoria, ya que la ejecución de las instrucciones no debe sufrir ningún retraso.

Unidad de Interfaz con el Bus

- Controla el intercambio de información entre el procesador y el exterior, a través de los buses.
- Lee continuamente el código del programa a solicitud de la *Unidad de Pre búsqueda*, y los almacena en una cola de instrucciones **FIFO** (**F**irst **I**n **F**irst **O**ut) de 16 bytes.
- Busca en memoria los operandos para las instrucciones de modo de direccionamiento indirecto a solicitud de la *Unidad de Ejecución*.
- El bloque de Priorización de Requerimientos de la **BIU**, decide ante requerimientos simultáneos cual será cursado antes.
- La búsqueda de Operandos tendrá la de más alta prioridad, y pospondrá cualquier otro acceso a memoria, ya que la ejecución de las instrucciones no debe sufrir ningún retraso.
- La dirección de memoria del operando o de la instrucción es requerida siempre a la **MMU**.

Unidad de Pre Búsqueda

Unidad de Pre Búsqueda

- Envía periódicamente a la BIU requerimientos de Opcode Fetch.

Unidad de Pre Búsqueda

- Envía periódicamente a la BIU requerimientos de Opcode Fetch.
- Administra la cola de pre búsqueda.

Unidad de Pre Búsqueda

- Envía periódicamente a la BIU requerimientos de Opcode Fetch.
- Administra la cola de pre búsqueda.
- La longitud promedio de una instrucción del 386 son 3,2 bytes, de modo que esta cola puede almacenar entre 4 y 5 instrucciones en promedio.

Unidad de Decodificación

Unidad de Decodificación

- La unidad de decodificación lee instrucciones de la cola de prebúsqueda y las decodifica.

Unidad de Decodificación

- La unidad de decodificación lee instrucciones de la cola de prebúsqueda y las decodifica.
- Mantiene una cola de pre-decodificación de hasta 3 palabras de 88 bits.

Unidad de Decodificación

- La unidad de decodificación lee instrucciones de la cola de prebúsqueda y las decodifica.
- Mantiene una cola de pre-decodificación de hasta 3 palabras de 88 bits.
- Recibe una señal de la Unidad de Ejecución que fuerza el ***flush*** del contenido de la cola de instrucciones.

Unidad de Ejecución

Unidad de Ejecución

- Contiene los registros de propósito general del procesador, registros temporarios (trasparentes al programador), y un registro rápido de 64 bits que permite acelerar gran parte de las instrucciones del 386.

Unidad de Ejecución

- Contiene los registros de propósito general del procesador, registros temporarios (trasparentes al programador), y un registro rápido de 64 bits que permite acelerar gran parte de las instrucciones del 386.

- La Unidad de Chequeo y Control se encarga de detectar violaciones a las reglas de seguridad del procesador cuando se configura en Modo Protegido.

Unidad de Ejecución

- Contiene los registros de propósito general del procesador, registros temporarios (trasparentes al programador), y un registro rápido de 64 bits que permite acelerar gran parte de las instrucciones del 386.

- La Unidad de Chequeo y Control se encarga de detectar violaciones a las reglas de seguridad del procesador cuando se configura en Modo Protegido.

- Cada vez que requiere acceder a memoria para traer un operando o para depositar un resultado, la Unidad de Ejecución envía una señal a la BIU, que como ya hemos dicho le dará la máxima prioridad de modo de no demorar la ejecución de la instrucción en curso.

Tratamiento de los saltos

Tratamiento de los saltos

- Un salto ***taken*** altera el flujo del programa en curso.

Tratamiento de los saltos

- Un salto ***taken*** altera el flujo del programa en curso.
- La Unidad de Ejecución ya no debe ejecutar las instrucciones que estaban en curo en el pipeline (sucesoras secuenciales del salto), sino las que están a partir de la dirección a la que lleva el salto.

Tratamiento de los saltos

- Un salto ***taken*** altera el flujo del programa en curso.
- La Unidad de Ejecución ya no debe ejecutar las instrucciones que estaban en curo en el pipeline (sucesoras secuenciales del salto), sino las que están a partir de la dirección a la que lleva el salto.
- Cada vez que una instrucción de salto resulta ***taken***, la Unidad de Ejecución envía señales de *flush* a la Unidad de Prebúsqueda, y a la Unidad de Decodificación, que forzarán el ***flush*** del contenido de la cola de instrucciones y de la cola de instrucciones pre-decodificadas, respectivamente.

Unidad de Gestión de Memoria

Unidad de Gestión de Memoria

- La MMU tiene por misión traducir las direcciones lógicas a direcciones físicas.

Unidad de Gestión de Memoria

- La MMU tiene por misión traducir las direcciones lógicas a direcciones físicas.
- (segmento:offset) $\longmapsto$número de 32 bits que saldrá por el bus de address para realizar el acceso.

Unidad de Gestión de Memoria

- La MMU tiene por misión traducir las direcciones lógicas a direcciones físicas.
- (segmento:offset) $\longmapsto$número de 32 bits que saldrá por el bus de address para realizar el acceso.
- La unidad de Segmentación siempre está activa. Traduce la dirección lógica en una dirección lineal, (dirección dentro de un espacio de hasta 4 Gbytes de direcciones consecutivas).

Unidad de Gestión de Memoria

- La MMU tiene por misión traducir las direcciones lógicas a direcciones físicas.
- (segmento:offset) $\longmapsto$ número de 32 bits que saldrá por el bus de address para realizar el acceso.
- La unidad de Segmentación siempre está activa. Traduce la dirección lógica en una dirección lineal, (dirección dentro de un espacio de hasta 4 Gbytes de direcciones consecutivas).
- La Unidad de Paginación por defecto se encuentra inactiva, y puede habilitarse únicamente cuando el procesador está trabajando en el modo protegido a través del bit **`CR0.PG`**.

Unidad de Gestión de Memoria

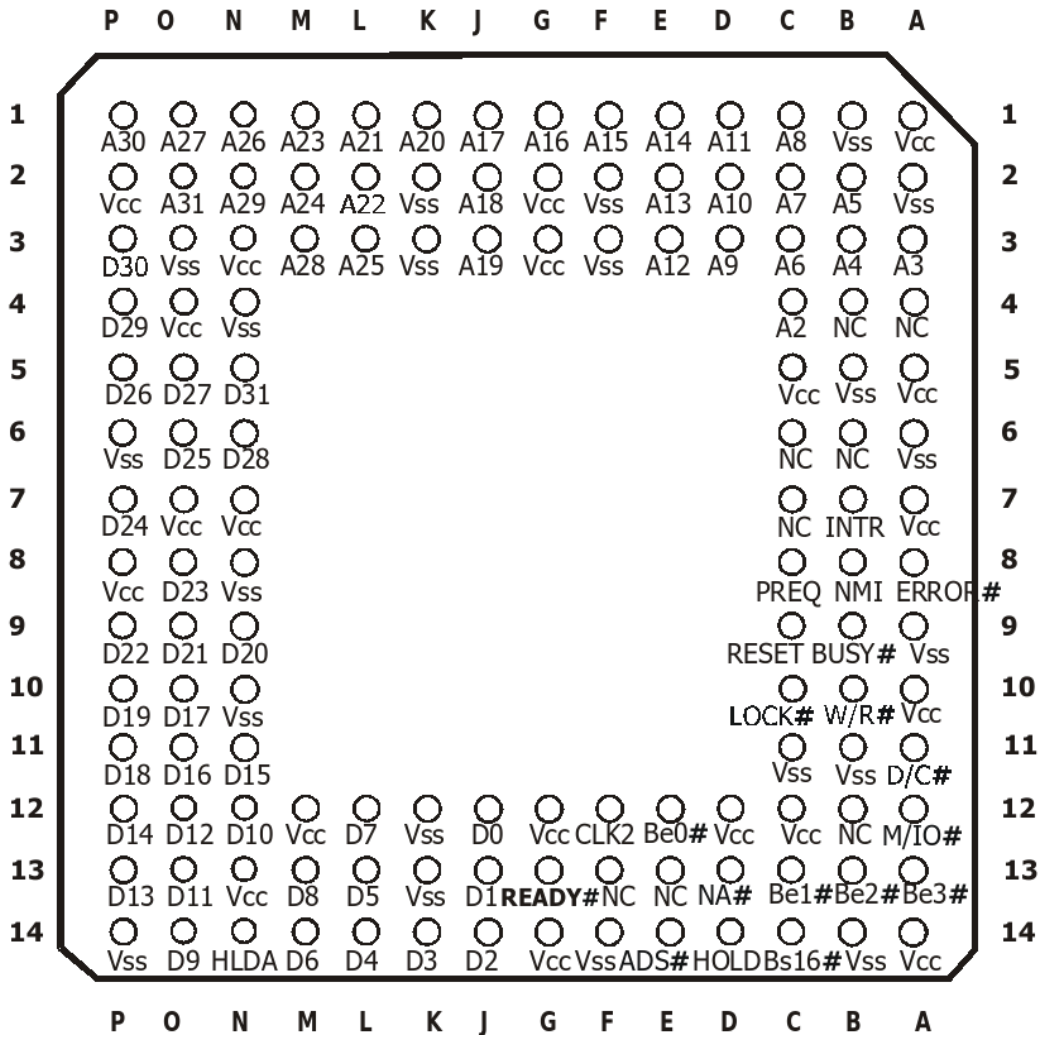
- La MMU tiene por misión traducir las direcciones lógicas a direcciones físicas.
- (segmento:offset) $\longmapsto$ número de 32 bits que saldrá por el bus de address para realizar el acceso.
- La unidad de Segmentación siempre está activa. Traduce la dirección lógica en una dirección lineal, (dirección dentro de un espacio de hasta 4 Gbytes de direcciones consecutivas).
- La Unidad de Paginación por defecto se encuentra inactiva, y puede habilitarse únicamente cuando el procesador está trabajando en el modo protegido a través del bit **`CR0.PG`**.
- Si está inactiva, la dirección física coincide con la dirección lineal generada por la Unidad de Segmentación.

Unidad de Gestión de Memoria

- La MMU tiene por misión traducir las direcciones lógicas a direcciones físicas.
- (segmento:offset) $\longmapsto$ número de 32 bits que saldrá por el bus de address para realizar el acceso.
- La unidad de Segmentación siempre está activa. Traduce la dirección lógica en una dirección lineal, (dirección dentro de un espacio de hasta 4 Gbytes de direcciones consecutivas).
- La Unidad de Paginación por defecto se encuentra inactiva, y puede habilitarse únicamente cuando el procesador está trabajando en el modo protegido a través del bit **`CR0.PG`**.
- Si está inactiva, la dirección física coincide con la dirección lineal generada por la Unidad de Segmentación.
- Si está activa, la dirección física se determina tomando el segmento definido y dividiéndolo en páginas consecutivas de 4 Kbytes.

Temario

Descripción de terminales del 80386 (Vista Superior)



Descripción de terminales del 80386 (Vista Superior)

El 80386 fue el primer procesador que introdujo el encapsulado PGA (Pin Grid Array) de 132 terminales que a semejanza de una matriz se determinan mediante coordenadas fila columna.
Por ejemplo la Señal **`W/R#`** se corresponde con la posición B10 en la vista superior del chip que se observa en la Figura del slide anterior.

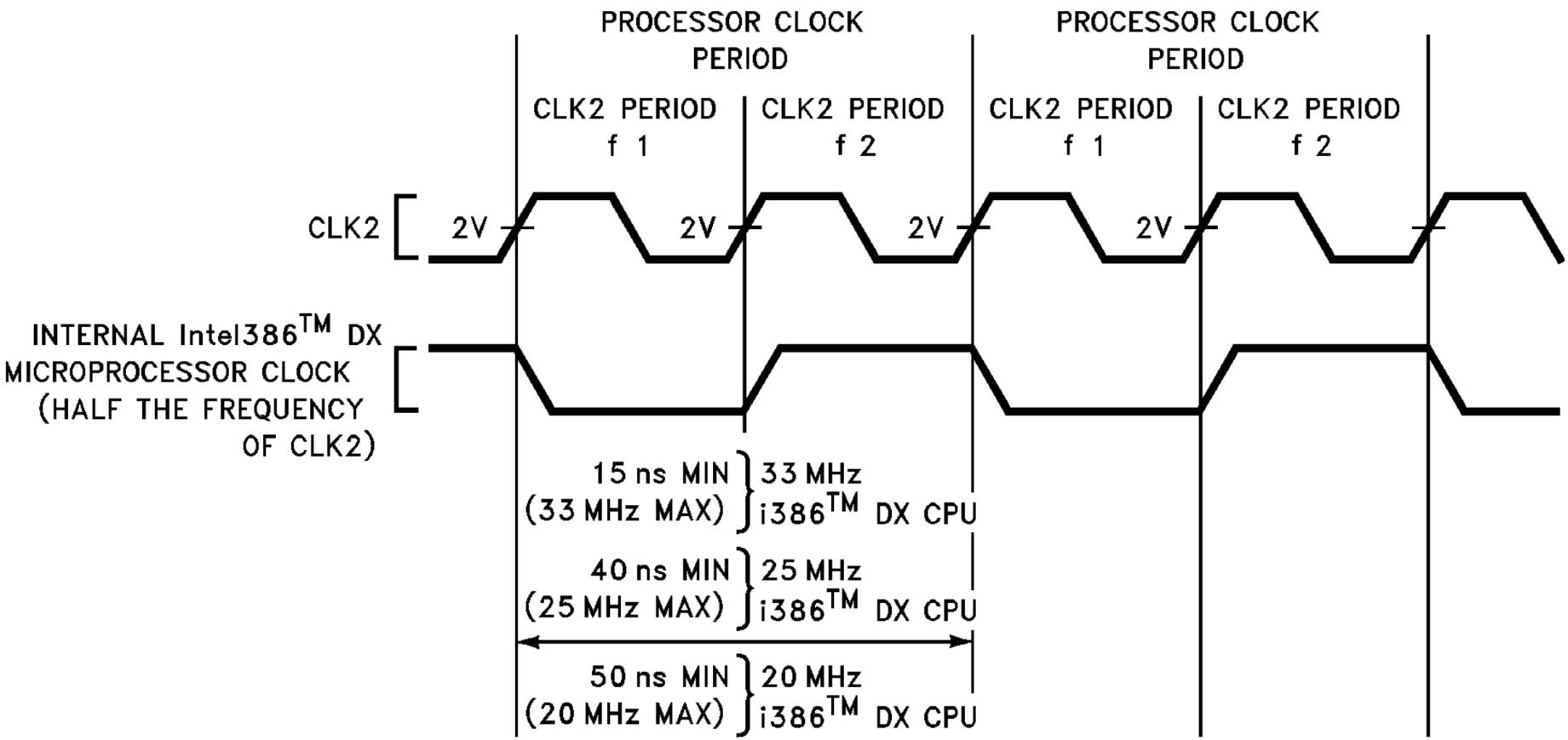
Alimentación

El 386 DX tiene 20 terminales para Vcc y 21 para Vss (o tierra). Deben conectarse a los niveles de tensión respectivos los 41 terminales, ya que por razones tecnológicas, (fundamentalmente para prevenir transitorios en la tensión de alimentación) el 386 alimenta por separado a diferentes partes de sus sustratos internos.
Vcc debe conectarse a una fuente de tensión de 5V ±5 % de corriente continua y 550 mA. de corriente máxima. A 20 Mhz., consume 460 mA.

Timing.

Internamente el 80386DX trabaja a la mitad de la frecuencia que le ingresa por su terminal **CLK2**. Sin embargo la exploración de alguna de las entradas del Bus de Control se realiza durante los flancos ascendentes de la señal **CLK2** y en algunos otros casos durante los flancos descendentes de dicha señal.
Sin embargo el tiempo mínimo en el que se puede realizar una operación elemental en el bus es un ciclo de CLK (o bien dos ciclos de **CLK2**). A este tiempo se lo denomina estado. Cada estado se divide en dos fases 1 y 2.

Timing.


PROCESSOR CLOCK PERIOD
PROCESSOR CLOCK PERIOD
CLK2 PERIOD f 1
CLK2 PERIOD f 2
CLK2 PERIOD f 1
CLK2 PERIOD f 2
CLK2
2V
2V
2V
2V
INTERNAL Intel386™ DX MICROPROCESSOR CLOCK (HALF THE FREQUENCY OF CLK2)
15 ns MIN (33 MHz MAX)
33 MHz i386™ DX CPU
40 ns MIN (25 MHz MAX)
25 MHz i386™ DX CPU
50 ns MIN (20 MHz MAX)
20 MHz i386™ DX CPU

Señales de Interrupción

Señales de Interrupción

INTR **INT**errupt Request. Señal de entrada activa alta por nivel y asincrónica mediante la cual recibe pedidos de interrupción desde los dispositivos periféricos. Trabaja con un controlador de interrupciones externo, que administra las solicitudes desde varios dispositivos. Es enmascarable, mediante un flag de la palabra de estados (`IF`).
Esta señal debe permanecer activa hasta recibir el reconocimiento del procesador, de lo contrario se pierde su petición (no se latchea).

Señales de Interrupción

INTR **INT**errupt Request. Señal de entrada activa alta por nivel y asincrónica mediante la cual recibe pedidos de interrupción desde los dispositivos periféricos. Trabaja con un controlador de interrupciones externo, que administra las solicitudes desde varios dispositivos. Es enmascarable, mediante un flag de la palabra de estados (`IF`).
Esta señal debe permanecer activa hasta recibir el reconocimiento del procesador, de lo contrario se pierde su petición (no se latchea).

NMI **N**on **M**askable **I**nterrupt. No puede enmascarase (es de alta prioridad). Corresponde al vector de interrupción 2. Se activa por flanco ascendente y en forma asincrónica. Debe permanecer en estado bajo mínimo 8 ciclos de la señal (`CLK2`) para asegurar su reconocimiento.
Una vez iniciada su rutina, no se interrumpe hasta que no se ejecute la instrucción IRET.

Bus de datos

El bus de datos del 80386DX está compuesto de 32 líneas bidireccionales three-state, denominadas D_0 a D_{31}. Se agrupan en cuatro bytes D_0-D_7, D_8-D_{15}, D_{16}-D_{23}, D_{24}-D_{31}.
El procesador puede configurar a 16 bits el "ancho"del bus de datos mediante la señal de entrada BS16#. Esto le permite adaptarse a dispositivos externos que no posean 32 bits datos para acceder al bus. De este modo soporta conexión con dispositivos de 16 bits de datos sin necesidad de agregar hardware adicional.

Bus de direcciones

Bus de direcciones

- A pesar de tener un bus de datos de 32 bits, el 386 organiza la memoria en bytes, debido a compatibilidad con los procesadores anteriores de 16 bits que, recordemos, nacieron con el 8088 cuyo bus de datos era de 8 bits.

Bus de direcciones

- A pesar de tener un bus de datos de 32 bits, el 386 organiza la memoria en bytes, debido a compatibilidad con los procesadores anteriores de 16 bits que, recordemos, nacieron con el 8088 cuyo bus de datos era de 8 bits.
- Ademas permite el acceso a datos de tipo *byte*, *word*, y *doble word* aunque éstos no estén alineados a doble word. En el caso de los procesadores previos de 16 bits, se aceptaba almacenar datos a partir de posiciones impares.

Bus de direcciones

- A pesar de tener un bus de datos de 32 bits, el 386 organiza la memoria en bytes, debido a compatibilidad con los procesadores anteriores de 16 bits que, recordemos, nacieron con el 8088 cuyo bus de datos era de 8 bits.
- Ademas permite el acceso a datos de tipo *byte*, *word*, y *doble word* aunque éstos no estén alineados a doble word. En el caso de los procesadores previos de 16 bits, se aceptaba almacenar datos a partir de posiciones impares.
- De este modo los procesadores de Intel permiten aprovechar la capacidad total de memoria. Al asumir esta política de organización de la memoria por bytes, se aprovecha el 100 % del espacio de memoria.

Bus de direcciones

Bus de direcciones

- Se compone de 30 líneas de salida three-state: A_2 a A_{31}.

Bus de direcciones

- Se compone de 30 líneas de salida three-state: A_2 a A_{31}.
- Es evidente la ausencia de las señales A_0 y A_1, que componen los dos bits menos significativos del bus de direcciones.

Bus de direcciones

- Se compone de 30 líneas de salida three-state: A_2 a A_{31}.
- Es evidente la ausencia de las señales A_0 y A_1, que componen los dos bits menos significativos del bus de direcciones.
- En su lugar el 80386 tiene 4 líneas auxiliares: **`BE0#`**, **`BE1#`**, **`BE2#`**, y **`BE3#`**.

Bus de direcciones

- Se compone de 30 líneas de salida three-state: A_2 a A_{31}.
- Es evidente la ausencia de las señales A_0 y A_1, que componen los dos bits menos significativos del bus de direcciones.
- En su lugar el 80386 tiene 4 líneas auxiliares: **`BE0#`**, **`BE1#`**, **`BE2#`**, y **`BE3#`**.
- A partir de éstas se pueden generar si se las necesita las señales A_0 y A_1 mediante una lógica simple de compuertas.

Bus de direcciones

- Se compone de 30 líneas de salida three-state: A_2 a A_{31}.
- Es evidente la ausencia de las señales A_0 y A_1, que componen los dos bits menos significativos del bus de direcciones.
- En su lugar el 80386 tiene 4 líneas auxiliares: **`BE0#`**, **`BE1#`**, **`BE2#`**, y **`BE3#`**.
- A partir de éstas se pueden generar si se las necesita las señales A_0 y A_1 mediante una lógica simple de compuertas.
- Las salidas **`BEn#`** indican cual/es de los cuatro bytes que componen el bus de datos actuarán en la transferencia en curso.

Bus de direcciones

- Se compone de 30 líneas de salida three-state: A_2 a A_{31}.
- Es evidente la ausencia de las señales A_0 y A_1, que componen los dos bits menos significativos del bus de direcciones.
- En su lugar el 80386 tiene 4 líneas auxiliares: **`BE0#`**, **`BE1#`**, **`BE2#`**, y **`BE3#`**.
- A partir de éstas se pueden generar si se las necesita las señales A_0 y A_1 mediante una lógica simple de compuertas.
- Las salidas **`BEn#`** indican cual/es de los cuatro bytes que componen el bus de datos actuarán en la transferencia en curso.
- De este modo, dividiendo la memoria principal del sistema (DRAM) en cuatro bancos y utilizando adecuadamente estas señales en la lógica de habilitación de dichos bancos de memoria, se aprovecha esta capacidad del 386.

Señales de estado del ciclo de Bus

Ciclo de Bus: Tiempo que transcurre para realizar una transferencia completa por el bus del sistema (lectura o escritura de Memoria o de E/S, Opcode Fetch, por citar las mas comunes).

Señales de estado del ciclo de Bus

Ciclo de Bus: Tiempo que transcurre para realizar una transferencia completa por el bus del sistema (lectura o escritura de Memoria o de E/S, Opcode Fetch, por citar las mas comunes).

M/IO# Memory/InputOutput#. Cuando está en '1' indica si la transacción es con Memoria, en otro caso la transacción es con Entrada Salida.

Señales de estado del ciclo de Bus

Ciclo de Bus: Tiempo que transcurre para realizar una transferencia completa por el bus del sistema (lectura o escritura de Memoria o de E/S, Opcode Fetch, por citar las mas comunes).

M/IO# Memory/InputOutput#. Cuando está en ’1’ indica si la transacción es con Memoria, en otro caso la transacción es con Entrada Salida.

W/R# Write/Read#. Indica la realización de un ciclo de escritura cuando se envía un ’1’, o de lectura se envía un ’0’.

Señales de estado del ciclo de Bus

Ciclo de Bus: Tiempo que transcurre para realizar una transferencia completa por el bus del sistema (lectura o escritura de Memoria o de E/S, Opcode Fetch, por citar las mas comunes).

M/IO# Memory/InputOutput#. Cuando está en ’1’ indica si la transacción es con Memoria, en otro caso la transacción es con Entrada Salida.

W/R# Write/Read#. Indica la realización de un ciclo de escritura cuando se envía un ’1’, o de lectura se envía un ’0’.

D/C# Data/Control#. Diferencia los ciclos de transferencia de datos (lectura o escritura de E/S o memoria) de los de control Interrupt Acknowledge, Halt, etc.).

Señales de estado del ciclo de Bus

Ciclo de Bus: Tiempo que transcurre para realizar una transferencia completa por el bus del sistema (lectura o escritura de Memoria o de E/S, Opcode Fetch, por citar las mas comunes).

M/IO# Memory/InputOutput#. Cuando está en '1' indica si la transacción es con Memoria, en otro caso la transacción es con Entrada Salida.

W/R# Write/Read#. Indica la realización de un ciclo de escritura cuando se envía un '1', o de lectura se envía un '0'.

D/C# Data/Control#. Diferencia los ciclos de transferencia de datos (lectura o escritura de E/S o memoria) de los de control Interrupt Acknowledge, Halt, etc.).

LOCK# Indica que el ciclo de bus en curso es un ciclo de bus atómico, es decir que no se suspende ni se cede el bus a ningún otro procesador o controlador de DMA por ejemplo, para que lo acceda durante este ciclo.

Estados del Bus

M/IO#	D/C#	W/R#	Tipos de ciclo de Bus		Bus Inhibido
0	0	0	Reconocimiento de Interrupción		Si
0	0	1	-		-
0	1	0	Lectura de E/S		No
0	1	1	Escritura de E/S		No
1	0	0	Lectura de código de Memoria		No
1	0	1	WAIT	SHUTDOWN	No
			Dirección = 2	Dirección = 0	
			BE0# = 1	BE0# = 0	
			BE1# = 1	BE1# = 1	
			BE2# = 0	BE2# = 1	
			BE3# = 1	BE3# = 1	
			A2-A31 = 0	A2-A31 = 0	
1	1	0	Lectura de datos de memoria		Algunos ciclos
1	1	1	Escritura de datos de memoria		Algunos ciclos

Nota;**`ADS#`** está activa. No interviene **`LOCK#`** ya que no tiene acción mas que como cerrojo del bus

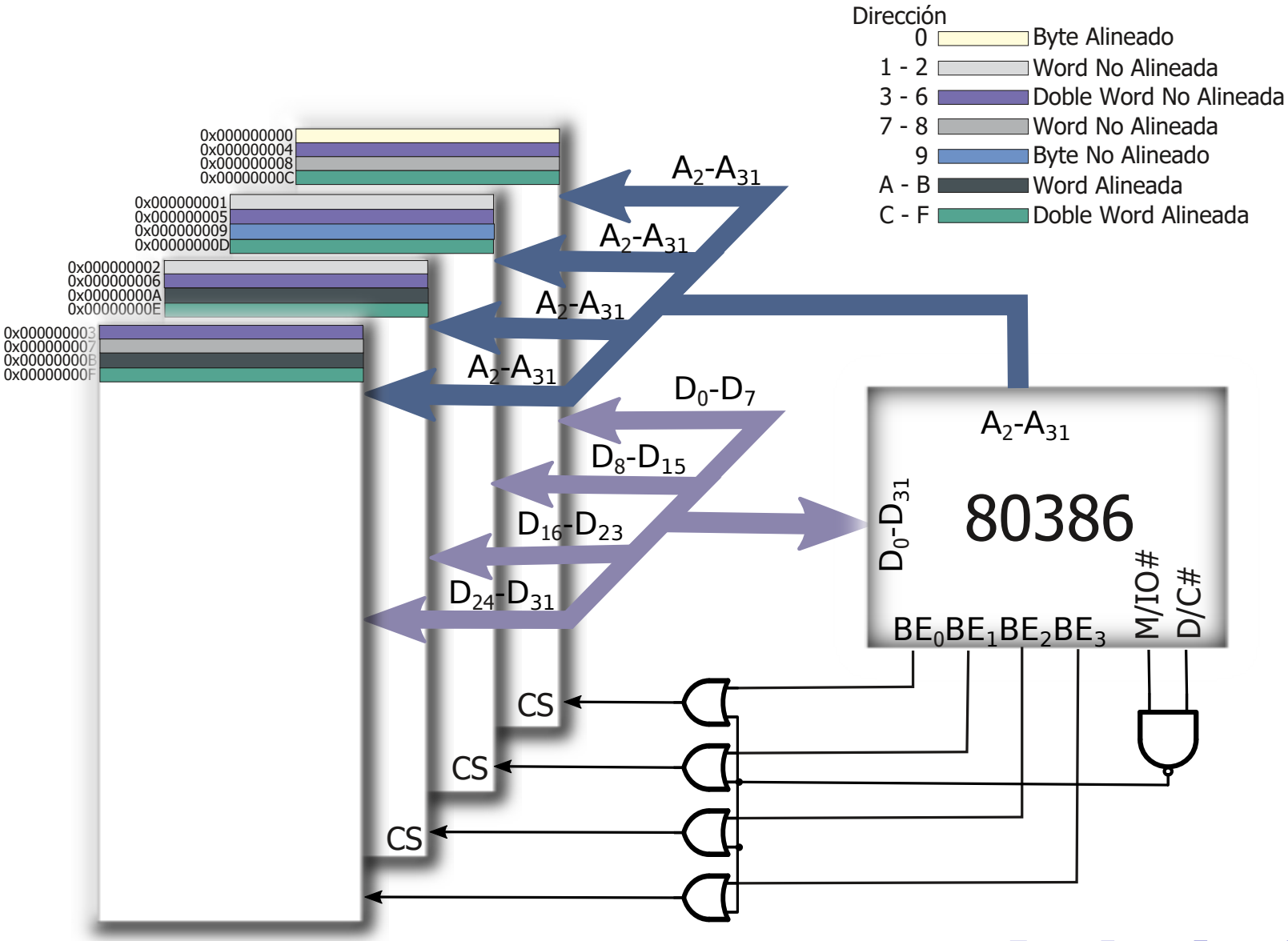
Organización de la memoria Física: BE0#-BE3#

La memoria se organiza en cuatro bancos que contienen direcciones distantes cada cuatro bytes:

- El primer banco contiene las direcciones 0x000000000, 0x00000004, 0x00000008, 0x0000000C, etc.
- El segundo las direcciones 0x00000001, 0x00000005, 0x00000009, 0x0000000D, etc.
- El tercero las direcciones 0x00000002, 0x00000006, 0x0000000A, 0x0000000E, etc.
- El cuarto las direcciones 0x00000003, 0x00000007, 0x0000000B, 0x0000000F, etc.

Luego cada señal **`BEn#`** interviene en la decodificación operando como Chip Select de cada banco de memoria.

Organización de la memoria Física: BE0#-BE3#



Inicio de un ciclo de Bus: ADS#

Inicio de un ciclo de Bus: ADS#

- El comienzo de un ciclo de bus es un atributo de un procesador. EL 80386, emplea al señal Address Status **ADS#**.

Inicio de un ciclo de Bus: ADS#

- El comienzo de un ciclo de bus es un atributo de un procesador. EL 80386, emplea al señal Address Status **ADS#**.
- Esta señal es activa baja e indica la validez de la dirección presente en las líneas $\mathbf{A_2}$-$\mathbf{A_{31}}$, **BE0#**, **BE1#**, **BE2#** y **BE3#**, mas las otras cuatro líneas de control que componen este grupo de señales de estado del ciclo de Bus: **W/R#**, **D/C#**, **M/IO#**, y **LOCK#**.

Inicio de un ciclo de Bus: ADS#

- El comienzo de un ciclo de bus es un atributo de un procesador. EL 80386, emplea al señal Address Status **ADS#**.
- Esta señal es activa baja e indica la validez de la dirección presente en las líneas $\mathbf{A_2}$-$\mathbf{A_{31}}$, **BE0#**, **BE1#**, **BE2#** y **BE3#**, mas las otras cuatro líneas de control que componen este grupo de señales de estado del ciclo de Bus: **W/R#**, **D/C#**, **M/IO#**, y **LOCK#**.
- Claramente, esta señal indica al exterior el comienzo de un ciclo de Bus.

Finalización de un ciclo de Bus: READY#

Finalización de un ciclo de Bus: READY#

- **READY#** es una señal de entrada que indica la finalización del ciclo de Bus en curso.

Finalización de un ciclo de Bus: **READY#**

- **READY#** es una señal de entrada que indica la finalización del ciclo de Bus en curso.
- En un ciclo de lectura o reconocimiento de interrupción, si **READY#** se activa, el procesador guarda en un latch la información presente en las líneas de datos.

Finalización de un ciclo de Bus: **READY#**

- **READY#** es una señal de entrada que indica la finalización del ciclo de Bus en curso.
- En un ciclo de lectura o reconocimiento de interrupción, si **READY#** se activa, el procesador guarda en un latch la información presente en las líneas de datos.
- En un ciclo de escritura, **READY#** activa permite al procesador asumir que la información que envió por el bus de datos ha sido tomada por el dispositivo direccionado.

Finalización de un ciclo de Bus: **READY#**

- **READY#** es una señal de entrada que indica la finalización del ciclo de Bus en curso.
- En un ciclo de lectura o reconocimiento de interrupción, si **READY#** se activa, el procesador guarda en un latch la información presente en las líneas de datos.
- En un ciclo de escritura, **READY#** activa permite al procesador asumir que la información que envió por el bus de datos ha sido tomada por el dispositivo direccionado.
- El dispositivo externo (memoria o periférico) es el responsable del manejo de la línea **READY#**.

Finalización de un ciclo de Bus: READY#

Finalización de un ciclo de Bus: READY#

- Cada ciclo de bus consta de dos estados.

Finalización de un ciclo de Bus: **READY#**

- Cada ciclo de bus consta de dos estados.
- En el primero se validan los datos y direcciones (**ADS#**).

Finalización de un ciclo de Bus: **READY#**

- Cada ciclo de bus consta de dos estados.
- En el primero se validan los datos y direcciones (**ADS#**).
- En el segundo se explora la línea **READY#**. Si se la encuentra en estado bajo se concluye el ciclo de bus, caso contrario, se inserta un estado inactivo (WAIT State). Este proceso se repite hasta que **READY#** haya sido activada.

Finalización de un ciclo de Bus: **READY#**

- Cada ciclo de bus consta de dos estados.
- En el primero se validan los datos y direcciones (**ADS#**).
- En el segundo se explora la línea **READY#**. Si se la encuentra en estado bajo se concluye el ciclo de bus, caso contrario, se inserta un estado inactivo (WAIT State). Este proceso se repite hasta que **READY#** haya sido activada.
- De este modo la comunicación entre el 386 y el mundo exterior es asincrónica, dado que un ciclo de bus comienza cuando se activa **ADS#** termina cuando se activa **READY#**.

Arbitraje del Bus

Handshake cuando el bus se comparte con otros procesadores.

Arbitraje del Bus

Handshake cuando el bus se comparte con otros procesadores.

HOLD Requerimiento del bus. Señal de entrada mediante la cual se solicita el bus al procesador. Debe permanecer activa (alta) durante todo el tiempo que dure la cesión del bus. Solo **RESET** es de mayor prioridad.

Arbitraje del Bus

Handshake cuando el bus se comparte con otros procesadores.

HOLD Requerimiento del bus. Señal de entrada mediante la cual se solicita el bus al procesador. Debe permanecer activa (alta) durante todo el tiempo que dure la cesión del bus. Solo **RESET** es de mayor prioridad.

HLDA Reconocimiento de requerimiento de bus. Mediante esta salida el 80386 comunica al exterior que cede el control del bus a otro Master. Es la respuesta al **HOLD** enviado previamente. Antes de activar esta señal el 80386 pasa al estado de alta impedancia las siguientes líneas: $\mathbf{D_0}$-$\mathbf{D_{31}}$, $\mathbf{A_2}$-$\mathbf{A_{31}}$, $\mathbf{BE_{0-3}}$#,**W/R#**, **D/C#**, **M/IO#**, **LOCK#**, **ADS#**. Si durante **HLDA** activa se produce una **NMI**, su estado se latchea para ser atendida ni bien se termine la cesión de bus, pero esta no es interrumpida (**NMI** es menos prioritaria).

Ciclos de Bus - Timing.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.
- Como ya se dijo un ciclo de Bus comienza al activarse la línea **ADS#**.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.
- Como ya se dijo un ciclo de Bus comienza al activarse la línea **ADS#**.
- En la tabla anterior se detallan los diferentes ciclos de bus posibles de acuerdo al estado de las líneas de control **M/IO#**, **W/R#**, y **D/C#**.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.
- Como ya se dijo un ciclo de Bus comienza al activarse la línea **ADS#**.
- En la tabla anterior se detallan los diferentes ciclos de bus posibles de acuerdo al estado de las líneas de control **M/IO#**, **W/R#**, y **D/C#**.
- El ciclo de bus **HALT** se produce como consecuencia de la ejecución de una instrucción **HLT**.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.
- Como ya se dijo un ciclo de Bus comienza al activarse la línea **ADS#**.
- En la tabla anterior se detallan los diferentes ciclos de bus posibles de acuerdo al estado de las líneas de control **M/IO#**, **W/R#**, y **D/C#**.
- El ciclo de bus **HALT** se produce como consecuencia de la ejecución de una instrucción **HLT**.
- **SHUTDOWN** se genera cuando se está procesando la excepción doble falta y se produce otra excepción de protección.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.
- Como ya se dijo un ciclo de Bus comienza al activarse la línea **ADS#**.
- En la tabla anterior se detallan los diferentes ciclos de bus posibles de acuerdo al estado de las líneas de control **M/IO#**, **W/R#**, y **D/C#**.
- El ciclo de bus **HALT** se produce como consecuencia de la ejecución de una instrucción **HLT**.
- **SHUTDOWN** se genera cuando se está procesando la excepción doble falta y se produce otra excepción de protección.
- Ambos ciclos son distinguibles desde el exterior por medio de las combinaciones de los estados de las líneas $\mathbf{A_2}$-$\mathbf{A_{31}}$, y $\mathbf{BE_{0-3}}$#.

Ciclos de Bus - Timing.

- Un ciclo de bus es el tiempo que demanda una transferencia completa y se compone de al menos de dos estados.
- Como ya se dijo un ciclo de Bus comienza al activarse la línea **ADS#**.
- En la tabla anterior se detallan los diferentes ciclos de bus posibles de acuerdo al estado de las líneas de control **M/IO#**, **W/R#**, y **D/C#**.
- El ciclo de bus **HALT** se produce como consecuencia de la ejecución de una instrucción **HLT**.
- **SHUTDOWN** se genera cuando se está procesando la excepción doble falta y se produce otra excepción de protección.
- Ambos ciclos son distinguibles desde el exterior por medio de las combinaciones de los estados de las líneas $\mathbf{A_2}$-$\mathbf{A_{31}}$, y $\mathbf{BE_{0-3}\#}$.
- En el caso de **HALT** se direcciona la posición de memoria 2, y en el caso de **SHUTDOWN** la 0.