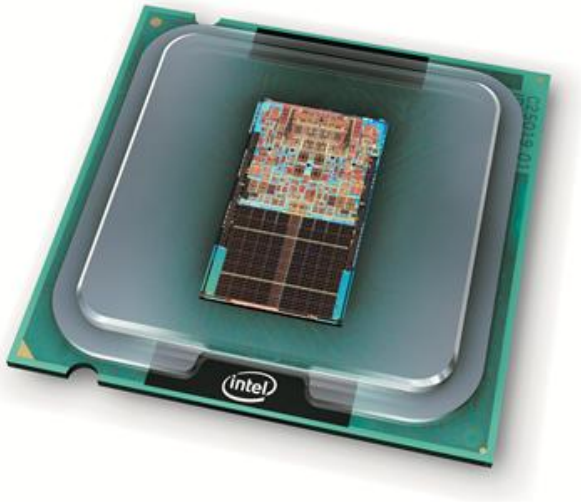


Organización del Computador II

Procesadores IA-32 e Intel® 64 - Inicialización. Boot de un SO

Alejandro Furfaro

Departamento de Computación - FCEyN UBA

26 de abril de 2018

Agenda

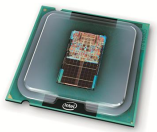
1. Inicialización de un computador IA-32 desde el Reset
   - Arranque de un Procesador BSP (o único)
2. Boot de un Sistema Operativo
   - Análisis Conceptual de la tarea
3. Modo Protegido
   - Responde a un requerimiento
   - Vamos a trabajar en Modo Protegido
4. Arrancando en 64 bits
   - Una píldora roja mas... “ancha”

1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro `eax` en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

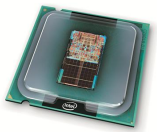


1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **eax** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

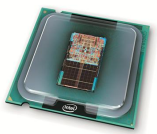


1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **eax** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

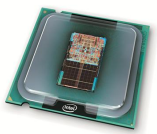


1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **eax** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

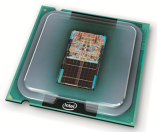


1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **eax** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

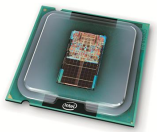
[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos



1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **`eax`** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

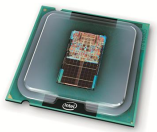


[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

1. Built In Self Test

- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **eax** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP

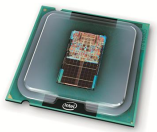


[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

1. Built In Self Test

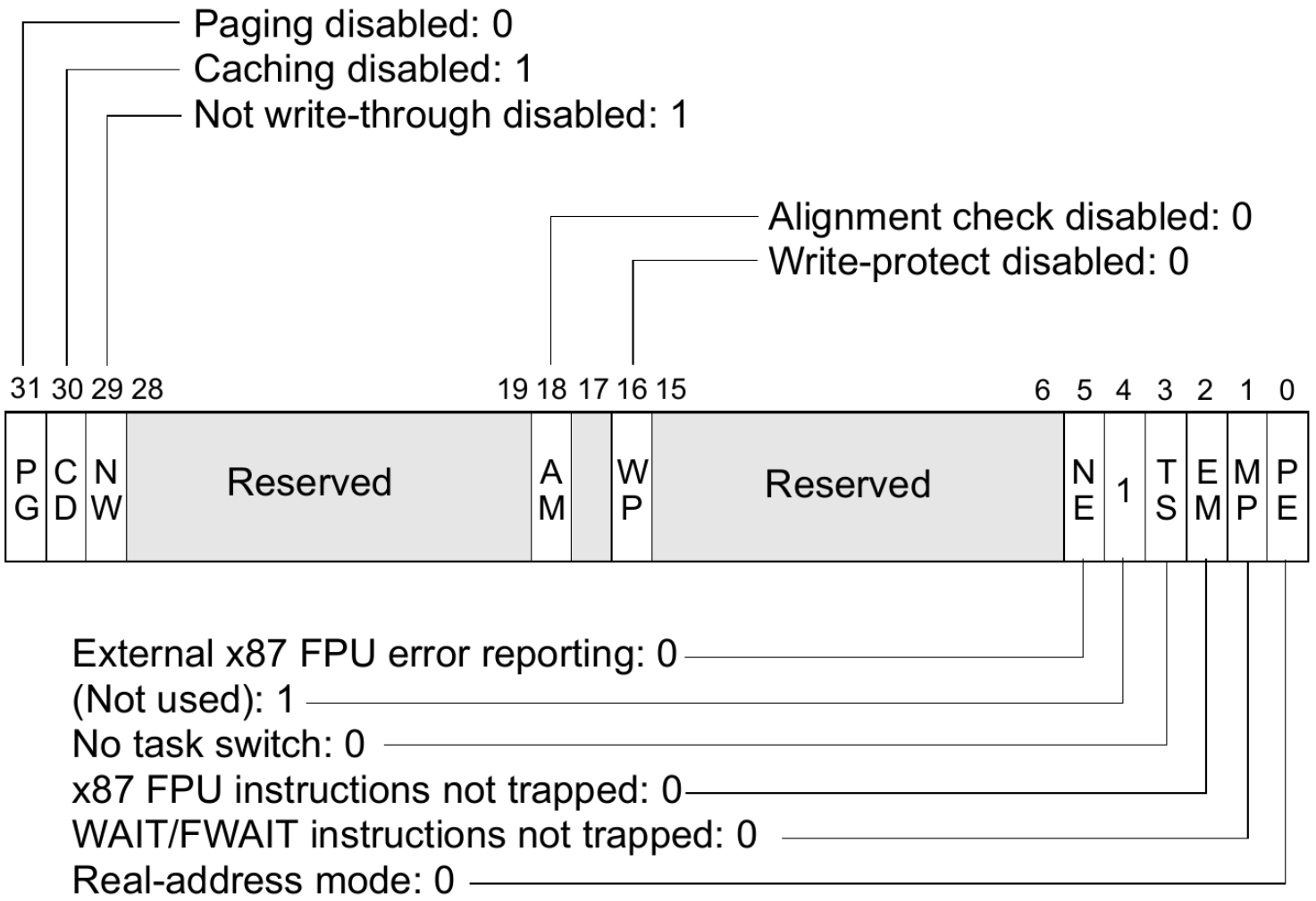
- Es un test interno que ejecuta el procesador luego de su encendido o cuando se activa su pin #RESET
- Como resultado inicializa el registro **eax** en 0x0 solo si todos los test pasaron OK.
- El resto de los registros toman un valor bien determinado
- Se invalidan las memorias cache internas
- Se invalidan las TLB[1]
- Se limpian los Branch Target Buffers[2]
- La diferencia con enviar una señal #INIT es que en éste caso, se mantienen intactos los valores de los Registros de la FPU, los MSR's y los MTRR's, y los caches internos.
- De acuerdo a la familia de procesador implementa con variantes la inicialización de los demás procesadores presentes en caso de un sistema SMP



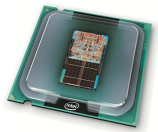
[1] Ver Paginación de Memoria

[2] Ver Microarquitectura - Predicción de saltos

2. Valores iniciales de interés



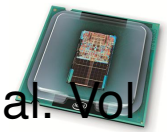
Registro CR0. Valor inicial



2. Valores iniciales de interés

Register	Power up	Reset	INIT
EFLAGS[1]	00000002H	00000002H	00000002H
EIP	0000FFF0H	0000FFF0H	0000FFF0H
CR0	60000010H[2]	60000010H[2]	60000010H[2]
CR2, CR3, CR4	00000000H	00000000H	00000000H
CS	Selector = F000H Base = FFFF0000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = F000H Base = FFFF0000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = F000H Base = FFFF0000H Limit = FFFFH AR = Present, R/W, Accessed
SS, DS, ES, FS, GS	Selector = 0000H Base = 00000000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = 0000H Base = 00000000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = 0000H Base = 00000000H Limit = FFFFH AR = Present, R/W, Accessed
EDX	000n06xxH[3]	000n06xxH[3]	000n06xxH[3]
EAX	0[4]	0[4]	0[4]
EBX, ECX, ESI, EDI, EBP, ESP	00000000H	00000000H	00000000H

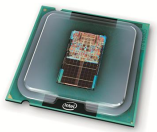
Valor iniciales de los registros corrientes[3]



[3] Para ampliar: Intel®64 and IA-32 Architectures Software Developer's Manual. Vol 3a. Capítulo 9

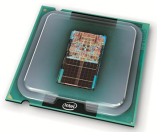
3.Reset Vector

- Cuando un procesador de Intel se enciende busca su primer instrucción en la dirección 0xFFFFFFF0.
- Esta dirección se conoce como **Reset Vector**
- Este acceso a un dispositivo Flash (NVRAM), es a 16 bytes de uno de los topes de operación de la memoria (4Gbyte).
- Todo lo que se puede colocar allí es una instrucción de salto



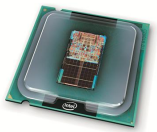
3.Reset Vector

- Cuando un procesador de Intel se enciende busca su primer instrucción en la dirección 0xFFFFFFF0.
- Esta dirección se conoce como **Reset Vector**
- Este acceso a un dispositivo Flash (NVRAM), es a 16 bytes de uno de los topes de operación de la memoria (4Gbyte).
- Todo lo que se puede colocar allí es una instrucción de salto



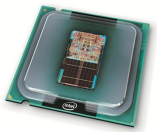
3.Reset Vector

- Cuando un procesador de Intel se enciende busca su primer instrucción en la dirección 0xFFFFFFF0.
- Esta dirección se conoce como **Reset Vector**
- Este acceso a un dispositivo Flash (NVRAM), es a 16 bytes de uno de los topes de operación de la memoria (4Gbyte).
- Todo lo que se puede colocar allí es una instrucción de salto



3.Reset Vector

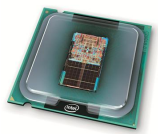
- Cuando un procesador de Intel se enciende busca su primer instrucción en la dirección 0xFFFFFFF0.
- Esta dirección se conoce como **Reset Vector**
- Este acceso a un dispositivo Flash (NVRAM), es a 16 bytes de uno de los topes de operación de la memoria (4Gbyte).
- Todo lo que se puede colocar allí es una instrucción de salto



3. Reset Vector

Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

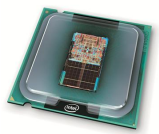
- **EIP** = 0x0000FFF0
- **CS** = 0xF000 <-*Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - Base = 0xFFFF0000
  - Límite = 0x0FFFF
  - Atrib = P=1,DPL=0,S=1,Tipo=10h,A=1
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



3. Reset Vector

Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

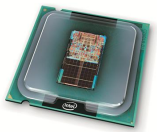
- **EIP** = 0x0000FFF0
- **CS** = 0xF000 <-*Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - Base = 0xFFFF0000
  - Límite = 0x0FFFF
  - Atrib = P=1,DPL=0x0,S=1,Tipo=1010b
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



3. Reset Vector

Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

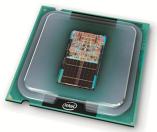
- **EIP** = 0x0000FFF0
- **CS** = 0xF000 *<-Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - **Base** = 0x**FFF**F0000
  - **Límite** = 0x0FFFF
  - **Atrib** = P=1:DPL=xx:S=1:Tipo=101:A=1
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



3. Reset Vector

Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

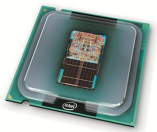
- **EIP** = 0x0000FFF0
- **CS** = 0xF000 *<-Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - **Base** = 0x**FFF**F0000
  - **Límite** = 0x0FFFF
  - **Atrib** = P=1:DPL=xx:S=1:Tipo=101:A=1
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



3. Reset Vector

Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

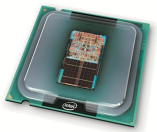
- **EIP** = 0x0000FFF0
- **CS** = 0xF000 *<-Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - **Base** = 0x**FFF**F0000
  - **Límite** = 0x0FFFF
  - **Atrib** = P=1:DPL=xx:S=1:Tipo=101:A=1
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



3. Reset Vector

Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

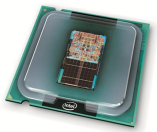
- **EIP** = 0x0000FFF0
- **CS** = 0xF000 *<-Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - **Base** = 0x**FFF**F0000
  - **Límite** = 0x0FFFF
  - **Atrib** = P=1:DPL=xx:S=1:Tipo=101:A=1
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



3. Reset Vector

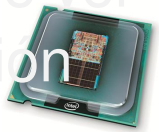
Cálculo de la dirección del reset vector en Modo Real, a partir de los valores iniciales

- **EIP** = 0x0000FFF0
- **CS** = 0xF000 *<-Esta es la parte visible*. Como veremos, **CS** tiene una parte oculta, que luego del reset toma los siguientes valores:
  - **Base** = 0x**FFF**F0000
  - **Límite** = 0x0FFFF
  - **Atrib** = P=1:DPL=xx:S=1:Tipo=101:A=1
- **Base** + **EIP** = 0xFFFF0000 + 0x0000FFF0 = **0xFFFFFFF0**



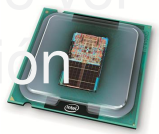
4. Selección de Modo

- En este punto es conveniente ocuparse de establecer el modo en que el procesador debe funcionar.
- Hay tres posibilidades.
  1. Dejarlo en modo Real
  2. Establecer Modo Protegido Flat
  3. Establecer Modo Protegido Segmentado (no recomendado para escribir firmware)
- Los modos 2 y 3 son variantes del Modo Protegido y tiene que ver con la forma en que organizaremos el modelo de segmentación



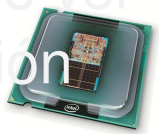
4. Selección de Modo

- En este punto es conveniente ocuparse de establecer el modo en que el procesador debe funcionar.
- Hay tres posibilidades.
  1. Dejarlo en modo Real
  2. Establecer Modo Protegido Flat
  3. Establecer Modo Protegido Segmentado (no recomendado para escribir firmware)
- Los modos 2 y 3 son variantes del Modo Protegido y tiene que ver con la forma en que organizaremos el modelo de segmentación



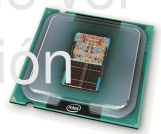
4. Selección de Modo

- En este punto es conveniente ocuparse de establecer el modo en que el procesador debe funcionar.
- Hay tres posibilidades.
  1. Dejarlo en modo Real
  2. Establecer Modo Protegido Flat
  3. Establecer Modo Protegido Segmentado (no recomendado para escribir firmware)
- Los modos 2 y 3 son variantes del Modo Protegido y tiene que ver con la forma en que organizaremos el modelo de segmentación



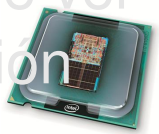
4. Selección de Modo

- En este punto es conveniente ocuparse de establecer el modo en que el procesador debe funcionar.
- Hay tres posibilidades.
  1. Dejarlo en modo Real
  2. Establecer Modo Protegido Flat
  3. Establecer Modo Protegido Segmentado (no recomendado para escribir firmware)
- Los modos 2 y 3 son variantes del Modo Protegido y tiene que ver con la forma en que organizaremos el modelo de segmentación



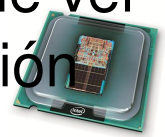
4. Selección de Modo

- En este punto es conveniente ocuparse de establecer el modo en que el procesador debe funcionar.
- Hay tres posibilidades.
  1. Dejarlo en modo Real
  2. Establecer Modo Protegido Flat
  3. Establecer Modo Protegido Segmentado (no recomendado para escribir firmware)
- Los modos 2 y 3 son variantes del Modo Protegido y tiene que ver con la forma en que organizaremos el modelo de segmentación



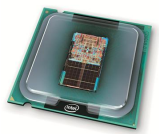
4. Selección de Modo

- En este punto es conveniente ocuparse de establecer el modo en que el procesador debe funcionar.
- Hay tres posibilidades.
  1. Dejarlo en modo Real
  2. Establecer Modo Protegido Flat
  3. Establecer Modo Protegido Segmentado (no recomendado para escribir firmware)
- Los modos 2 y 3 son variantes del Modo Protegido y tiene que ver con la forma en que organizaremos el modelo de segmentación



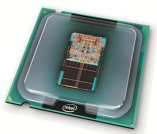
3. Selección de Modo

- Cuando el procesador se enciende, o luego de activarse su terminal de #RESET o #INIT, está indefectiblemente en *Modo Real*.
- En este modo no puede direccionar mas de 1 Mbyte de memoria (delicias de la compatibilidad),
- Sin embargo el vector de Reset está en 0xFFFFFFF0.
- Por eso para poder fetchear allí su primer instrucción se resolvió que por los 12 líneas mas significativas del bus de Address (A_{20} a A_{31}), envíen siempre `1's`, es decir, aplicarle a la dirección resultante del Modo Real la máscara 0xFFFxxxxx (utilizando el valor de inicio en el **Registro Base** asociado al registro **CS**).
- Este comportamiento se mantendrá mientras no se ejecute un `jmp far`, un `call far`, u ocurra una interrupción.



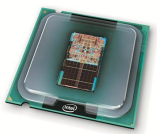
3. Selección de Modo

- Cuando el procesador se enciende, o luego de activarse su terminal de #RESET o #INIT, está indefectiblemente en *Modo Real*.
- En este modo no puede direccionar mas de 1 Mbyte de memoria (delicias de la compatibilidad),
- Sin embargo el vector de Reset está en 0xFFFFFFF0.
- Por eso para poder fetchear allí su primer instrucción se resolvió que por los 12 líneas mas significativas del bus de Address (A_{20} a A_{31}), envíen siempre `1's`, es decir, aplicarle a la dirección resultante del Modo Real la máscara 0xFFFxxxxx (utilizando el valor de inicio en el **Registro Base** asociado al registro **CS**).
- Este comportamiento se mantendrá mientras no se ejecute un `jmp far`, un `call far`, u ocurra una interrupción.



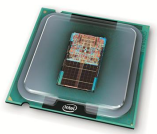
3. Selección de Modo

- Cuando el procesador se enciende, o luego de activarse su terminal de #RESET o #INIT, está indefectiblemente en *Modo Real*.
- En este modo no puede direccionar mas de 1 Mbyte de memoria (delicias de la compatibilidad),
- Sin embargo el vector de Reset está en 0xFFFFFFF0.
- Por eso para poder fetchear allí su primer instrucción se resolvió que por los 12 líneas mas significativas del bus de Address (A_{20} a A_{31}), envíen siempre `1's`, es decir, aplicarle a la dirección resultante del Modo Real la máscara 0xFFFxxxxx (utilizando el valor de inicio en el **Registro Base** asociado al registro **CS**).
- Este comportamiento se mantendrá mientras no se ejecute un `jmp far`, un `call far`, u ocurra una interrupción.



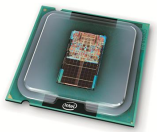
3. Selección de Modo

- Cuando el procesador se enciende, o luego de activarse su terminal de #RESET o #INIT, está indefectiblemente en *Modo Real*.
- En este modo no puede direccionar mas de 1 Mbyte de memoria (delicias de la compatibilidad),
- Sin embargo el vector de Reset está en 0xFFFFFFF0.
- Por eso para poder fetchear allí su primer instrucción se resolvió que por los 12 líneas mas significativas del bus de Address (A_{20} a A_{31}), envíen siempre `1's`, es decir, aplicarle a la dirección resultante del Modo Real la máscara 0xFFFxxxxx (utilizando el valor de inicio en el **Registro Base** asociado al registro **CS**).
- Este comportamiento se mantendrá mientras no se ejecute un `jmp far`, un `call far`, u ocurra una interrupción.



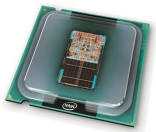
3. Selección de Modo

- Cuando el procesador se enciende, o luego de activarse su terminal de #RESET o #INIT, está indefectiblemente en *Modo Real*.
- En este modo no puede direccionar mas de 1 Mbyte de memoria (delicias de la compatibilidad),
- Sin embargo el vector de Reset está en 0xFFFFFFF0.
- Por eso para poder fetchear allí su primer instrucción se resolvió que por los 12 líneas mas significativas del bus de Address (A_{20} a A_{31}), envíen siempre `1's`, es decir, aplicarle a la dirección resultante del Modo Real la máscara 0xFFFxxxxx (utilizando el valor de inicio en el **Registro Base** asociado al registro **CS**).
- Este comportamiento se mantendrá mientras no se ejecute un `jmp far`, un `call far`, u ocurra una interrupción.



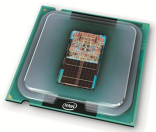
3. Selección de Modo

- Si estando en Modo Real se ejecuta **`jmp far`**, **`call far`**, u ocurre una interrupción, entonces el procesador por sí solo no puede acceder por encima del 1er. Mbyte de memoria RAM.
- Para poder seguir accediendo al espacio de direcciones en el que se encuentra mapeada la NVRAM, se requiere que el procesador trabaje con un chipset capaz de implementar aliasing de direcciones.
- Si el chipset tiene capacidad de aliasing genera a partir de direcciones de modo real, las direcciones en las que está mapeada la NVRAM cerca del tope de 4 Gbytes.



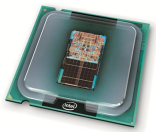
3. Selección de Modo

- Si estando en Modo Real se ejecuta **`jmp far`**, **`call far`**, u ocurre una interrupción, entonces el procesador por sí solo no puede acceder por encima del 1er. Mbyte de memoria RAM.
- Para poder seguir accediendo al espacio de direcciones en el que se encuentra mapeada la NVRAM, se requiere que el procesador trabaje con un chipset capaz de implementar aliasing de direcciones.
- Si el chipset tiene capacidad de aliasing genera a partir de direcciones de modo real, las direcciones en las que está mapeada la NVRAM cerca del tope de 4 Gbytes.



3. Selección de Modo

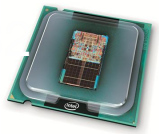
- Si estando en Modo Real se ejecuta **`jmp far`**, **`call far`**, u ocurre una interrupción, entonces el procesador por sí solo no puede acceder por encima del 1er. Mbyte de memoria RAM.
- Para poder seguir accediendo al espacio de direcciones en el que se encuentra mapeada la NVRAM, se requiere que el procesador trabaje con un chipset capaz de implementar aliasing de direcciones.
- Si el chipset tiene capacidad de aliasing genera a partir de direcciones de modo real, las direcciones en las que está mapeada la NVRAM cerca del tope de 4 Gbytes.



4. Preparar la Inicialización de memoria

- Todo el código que se ha ejecutado hasta este momento, se ha ejecutado desde NVRAM o sea Flash.
- Cuanto antes pasemos a ejecutar desde memoria DRAM, antes aceleraremos el inicio del sistema ya que la memoria DRAM es mas veloz que la Flash.
- Entonces es urgente inicializar la DRAM. Para ello hay que:
  1. Actualizar el microcódigo del procesador[1] (opcional)
  2. Inicializar el soporte del procesador para manejo de memoria
  3. Inicializar el chipset

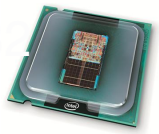
[1] Ampliar desde Section 9.11 Microcode Update Facilities, Intel 64 and IA-32 Architectures Software Developer's Manual Vol 3a



4. Preparar la Inicialización de memoria

- Todo el código que se ha ejecutado hasta este momento, se ha ejecutado desde NVRAM o sea Flash.
- Cuanto antes pasemos a ejecutar desde memoria DRAM, antes aceleraremos el inicio del sistema ya que la memoria DRAM es mas veloz que la Flash.
- Entonces es urgente inicializar la DRAM. Para ello hay que:
  1. Actualizar el microcódigo del procesador[4] (opcional)
  2. Inicializar el soporte del procesador para manejo de memoria
  3. Inicializar el chipset

[4] Ampliar desde Seccion 9.11 Microcode Update Facilities, Intel®64 and IA-32 Architectures Software Developer's Manual. Vol 3a.

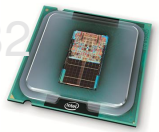


4. Preparar la Inicialización de memoria

- Todo el código que se ha ejecutado hasta este momento, se ha ejecutado desde NVRAM o sea Flash.
- Cuanto antes pasemos a ejecutar desde memoria DRAM, antes aceleraremos el inicio del sistema ya que la memoria DRAM es mas veloz que la Flash.
- Entonces es urgente inicializar la DRAM. Para ello hay que:
  1. Actualizar el microcódigo del procesador[4] (opcional)
  2. Inicializar el soporte del procesador para manejo de memoria[5]
  3. Inicializar el chipset[5]

[4] Ampliar desde Seccion 9.11 Microcode Update Facilities, Intel®64 and IA-32 Architectures Software Developer's Manual. Vol 3a.

[5] BIOS/Firmware Writer's Guide (a Black Magic Manual)



4. Preparar la Inicialización de memoria

- Todo el código que se ha ejecutado hasta este momento, se ha ejecutado desde NVRAM o sea Flash.
- Cuanto antes pasemos a ejecutar desde memoria DRAM, antes aceleraremos el inicio del sistema ya que la memoria DRAM es mas veloz que la Flash.
- Entonces es urgente inicializar la DRAM. Para ello hay que:
  1. Actualizar el microcódigo del procesador[4] (opcional)
  2. Inicializar el soporte del procesador para manejo de memoria[5]
  3. Inicializar el chipset[5]

[4] Ampliar desde Seccion 9.11 Microcode Update Facilities, Intel®64 and IA-32 Architectures Software Developer's Manual. Vol 3a.

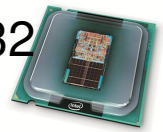
[5] BIOS/Firmware Writer's Guide (a Black Magic Manual)

4. Preparar la Inicialización de memoria

- Todo el código que se ha ejecutado hasta este momento, se ha ejecutado desde NVRAM o sea Flash.
- Cuanto antes pasemos a ejecutar desde memoria DRAM, antes aceleraremos el inicio del sistema ya que la memoria DRAM es mas veloz que la Flash.
- Entonces es urgente inicializar la DRAM. Para ello hay que:
  1. Actualizar el microcódigo del procesador[4] (opcional)
  2. Inicializar el soporte del procesador para manejo de memoria[5]
  3. Inicializar el chipset[5]

[4] Ampliar desde Seccion 9.11 Microcode Update Facilities, Intel®64 and IA-32 Architectures Software Developer's Manual. Vol 3a.

[5] BIOS/Firmware Writer's Guide (a Black Magic Manual)

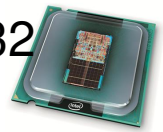


4. Preparar la Inicialización de memoria

- Todo el código que se ha ejecutado hasta este momento, se ha ejecutado desde NVRAM o sea Flash.
- Cuanto antes pasemos a ejecutar desde memoria DRAM, antes aceleraremos el inicio del sistema ya que la memoria DRAM es mas veloz que la Flash.
- Entonces es urgente inicializar la DRAM. Para ello hay que:
  1. Actualizar el microcódigo del procesador[4] (opcional)
  2. Inicializar el soporte del procesador para manejo de memoria[5]
  3. Inicializar el chipset[5]

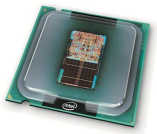
[4] Ampliar desde Seccion 9.11 Microcode Update Facilities, Intel®64 and IA-32 Architectures Software Developer's Manual. Vol 3a.

[5] BIOS/Firmware Writer's Guide (a Black Magic Manual)



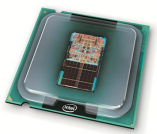
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory Initialization Reference Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



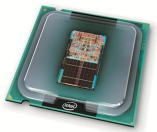
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory Initialization Reference Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



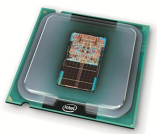
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory initialization Referencce Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



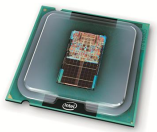
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory initialization Referencce Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



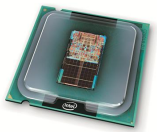
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory initialization Referencce Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



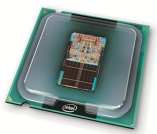
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory initialization Referencce Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



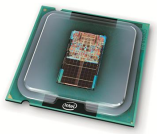
5. Inicialización de Memoria

- Otra caja negra.
- Intel provee (NDA mediante) los siguientes recursos:
  1. BIOS/Firmware Writer's Guide
  2. Memory initialization Referencce Code (MRC)
- Es chipset (plataforma) dependiente
- MRC puede estar escrito para ejecutar en código de 16 bits o 32 bits (Segmentación Flat o multisegmento).



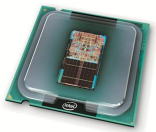
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



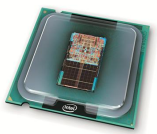
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Translation Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



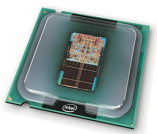
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



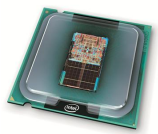
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



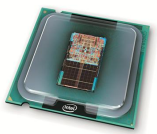
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



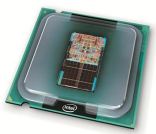
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



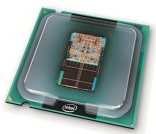
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



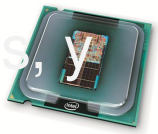
6. Inicialización Post Memoria

- Una vez ejecutado el código que inicializa el controlador de memoria y los configura de acuerdo a los DIMMs detectados en el sistema,se termina el trabajo de inicialización de memoria.
- Es una secuencia de pasos adicional, pero hecha por nosotros
  1. Memory test
  2. Shadow Firmware
  3. Memory Transaction Re-Direction
  4. Configurar un Stack
  5. Transferir control de programa a DRAM



6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, y pueden hacerse en assembler.

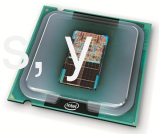


6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, y pueden hacerse en assembler.

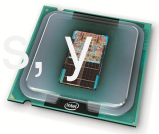
6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, y pueden hacerse en assembler.



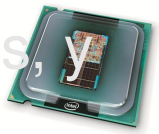
6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, y pueden hacerse en assembler.



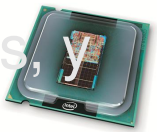
6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, y pueden hacerse en assembler.



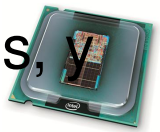
6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, pueden hacerse en assembler.



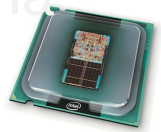
6.1. Memory Test

- Es un primer uso de la memoria para comprobar su integridad una vez zinicializado el controlador lo cual nos permite el acceso
- Puede estar incluido en el MRC (algunos vendors lo proveen)
- Es conveniente chequear memoria en este punto ya que los errores se producen de manera aleatoria e inconsistente.
- Simplifica el debug del firmware ya que si pasó el test, los errores posteriores no son atribuibles a memoria.
- Los test son un balance entre la granularidad (cada cuantos bytes se comprueba) vs. performance (en móviles el tiempo de arranque es importante)
- No hay algoritmos específicos, sino que son mas bien simples, y pueden hacerse en assembler.



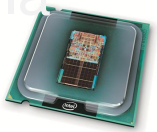
6.2. Firmware Shadow

- Es código que se autocopia desde NVRAM (Lenta, típicamente es FLASH) a DRAM (mas rápida y por lo tanto mejora el tiempo del setup.
- Al habilitar el cache, cada acceso a la Flash (NVRAM) genera un Miss y retrasa enormemente la ejecución.
- Copiarlo a DRAM no solo acelera por ser esta mas rápida que la NVRAM, sino por ser cacheable.
- En los sistemas PC la zona de shadow debe copiarse desde la NVRAM hasta la zona por debajo del Mbyte.
- En otros sistemas el espacio de direccionamiento destino de la copia es arbitraria.



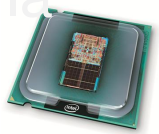
6.2. Firmware Shadow

- Es código que se autocopia desde NVRAM (Lenta, típicamente es FLASH) a DRAM (mas rápida y por lo tanto mejora el tiempo del setup.
- Al habilitar el cache, cada acceso a la Flash (NVRAM) genera un Miss y retrasa enormemente la ejecución.
- Copiarlo a DRAM no solo acelera por ser esta mas rápida que la NVRAM, sino por ser cacheable.
- En los sistemas PC la zona de shadow debe copiarse desde la NVRAM hasta la zona por debajo del Mbyte.
- En otros sistemas el espacio de direccionamiento destino de la copia es arbitraria.



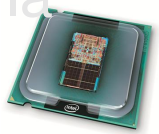
6.2. Firmware Shadow

- Es código que se autocopia desde NVRAM (Lenta, típicamente es FLASH) a DRAM (mas rápida y por lo tanto mejora el tiempo del setup.
- Al habilitar el cache, cada acceso a la Flash (NVRAM) genera un Miss y retrasa enormemente la ejecución.
- Copiarlo a DRAM no solo acelera por ser esta mas rápida que la NVRAM, sino por ser cacheable.
- En los sistemas PC la zona de shadow debe copiarse desde la NVRAM hasta la zona por debajo del Mbyte.
- En otros sistemas el espacio de direccionamiento destino de la copia es arbitraria.



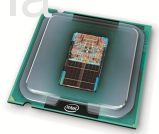
6.2. Firmware Shadow

- Es código que se autocopia desde NVRAM (Lenta, típicamente es FLASH) a DRAM (mas rápida y por lo tanto mejora el tiempo del setup.
- Al habilitar el cache, cada acceso a la Flash (NVRAM) genera un Miss y retrasa enormemente la ejecución.
- Copiarlo a DRAM no solo acelera por ser esta mas rápida que la NVRAM, sino por ser cacheable.
- En los sistemas PC la zona de shadow debe copiarse desde la NVRAM hasta la zona por debajo del Mbyte.
- En otros sistemas el espacio de direccionamiento destino de la copia es arbitraria.



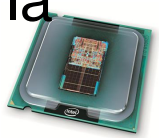
6.2. Firmware Shadow

- Es código que se autocopia desde NVRAM (Lenta, típicamente es FLASH) a DRAM (mas rápida y por lo tanto mejora el tiempo del setup.
- Al habilitar el cache, cada acceso a la Flash (NVRAM) genera un Miss y retrasa enormemente la ejecución.
- Copiarlo a DRAM no solo acelera por ser esta mas rápida que la NVRAM, sino por ser cacheable.
- En los sistemas PC la zona de shadow debe copiarse desde la NVRAM hasta la zona por debajo del Mbyte.
- En otros sistemas el espacio de direccionamiento destino de la copia es arbitraria.



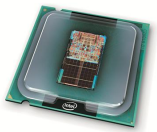
6.2. Firmware Shadow

- Es código que se autocopia desde NVRAM (Lenta, típicamente es FLASH) a DRAM (mas rápida y por lo tanto mejora el tiempo del setup.
- Al habilitar el cache, cada acceso a la Flash (NVRAM) genera un Miss y retrasa enormemente la ejecución.
- Copiarlo a DRAM no solo acelera por ser esta mas rápida que la NVRAM, sino por ser cacheable.
- En los sistemas PC la zona de shadow debe copiarse desde la NVRAM hasta la zona por debajo del Mbyte.
- En otros sistemas el espacio de direccionamiento destino de la copia es arbitraria.



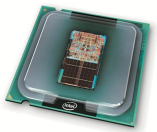
6.2. Memory Transaction Re-Direction

- Los chipsets generalmente están provistos de Registros PAM (Programmable Attribute Maps)
- Estos PAMs permiten controlar el aliasing de direcciones que hace posible leer o escribir secciones de memoria por debajo de 1 Mbyte hacia o desde DRAM o NVRAM cuyas direcciones rondan los 4 Gbytes.
- Antes de ejecutar shadowing puede requerirse acceder a estos registros.
- Los accesos dependen de cada chipset. Algunos permiten leer y escribir, otros solo leer.



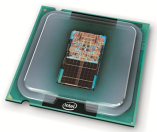
6.2. Memory Transaction Re-Direction

- Los chipsets generalmente están provistos de Registros PAM (Programmable Attribute Maps)
- Estos PAMs permiten controlar el aliasing de direcciones que hace posible leer o escribir secciones de memoria por debajo de 1 Mbyte hacia o desde DRAM o NVRAM cuyas direcciones rondan los 4 Gbytes.
- Antes de ejecutar shadowing puede requerirse acceder a estos registros.
- Los accesos dependen de cada chipset. Algunos permiten leer y escribir, otros solo leer.



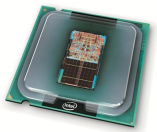
6.2. Memory Transaction Re-Direction

- Los chipsets generalmente están provistos de Registros PAM (Programmable Attribute Maps)
- Estos PAMs permiten controlar el aliasing de direcciones que hace posible leer o escribir secciones de memoria por debajo de 1 Mbyte hacia o desde DRAM o NVRAM cuyas direcciones rondan los 4 Gbytes.
- Antes de ejecutar shadowing puede requerirse acceder a estos registros.
- Los accesos dependen de cada chipset. Algunos permiten leer y escribir, otros solo leer.



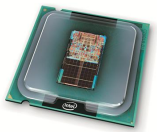
6.2. Memory Transaction Re-Direction

- Los chipsets generalmente están provistos de Registros PAM (Programmable Attribute Maps)
- Estos PAMs permiten controlar el aliasing de direcciones que hace posible leer o escribir secciones de memoria por debajo de 1 Mbyte hacia o desde DRAM o NVRAM cuyas direcciones rondan los 4 Gbytes.
- Antes de ejecutar shadowing puede requerirse acceder a estos registros.
- Los accesos dependen de cada chipset. Algunos permiten leer y escribir, otros solo leer.



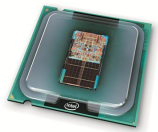
6.2. Memory Transaction Re-Direction

- Los chipsets generalmente están provistos de Registros PAM (Programmable Attribute Maps)
- Estos PAMs permiten controlar el aliasing de direcciones que hace posible leer o escribir secciones de memoria por debajo de 1 Mbyte hacia o desde DRAM o NVRAM cuyas direcciones rondan los 4 Gbytes.
- Antes de ejecutar shadowing puede requerirse acceder a estos registros.
- Los accesos dependen de cada chipset. Algunos permiten leer y escribir, otros solo leer.



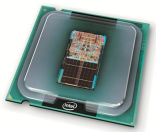
6.3. Establecimiento de un stack

- Antes de saltar a memoria es necesario definir un stack
- Es necesario definir la cantidad de memoria y considerar que el stack crece hacia direcciones bajas.
- En modo 16 bits se configuran los registros `ss:sp`
- En modo Protegido (ya sea Flat o Segmentado) se configuran `ss:esp`



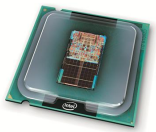
6.3. Establecimiento de un stack

- Antes de saltar a memoria es necesario definir un stack
- Es necesario definir la cantidad de memoria y considerar que el stack crece hacia direcciones bajas.
- En modo 16 bits se configuran los registros `ss:sp`
- En modo Protegido (ya sea Flat o Segmentado) se configuran `ss:esp`



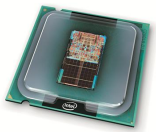
6.3. Establecimiento de un stack

- Antes de saltar a memoria es necesario definir un stack
- Es necesario definir la cantidad de memoria y considerar que el stack crece hacia direcciones bajas.
- En modo 16 bits se configuran los registros `ss:sp`
- En modo Protegido (ya sea Flat o Segmentado) se configuran `ss:esp`



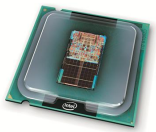
6.3. Establecimiento de un stack

- Antes de saltar a memoria es necesario definir un stack
- Es necesario definir la cantidad de memoria y considerar que el stack crece hacia direcciones bajas.
- En modo 16 bits se configuran los registros **`ss:sp`**
- En modo Protegido (ya sea Flat o Segmentado) se configuran **`ss:esp`**



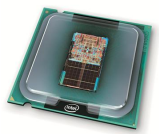
6.3. Establecimiento de un stack

- Antes de saltar a memoria es necesario definir un stack
- Es necesario definir la cantidad de memoria y considerar que el stack crece hacia direcciones bajas.
- En modo 16 bits se configuran los registros **`ss:sp`**
- En modo Protegido (ya sea Flat o Segmentado) se configuran **`ss:esp`**



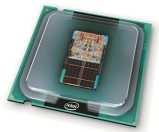
6.4. Salto a memoria DRAM

- Aquí es donde se puede producir un problema si no se hicieron bien las cosas.
- El programa debe haberse autocopiado de NVRAM a DRAM
- El salto se implementa mediante un salto FAR.
- En modo 16 bits o real hay que saltar a una dirección dentro del primer Mbyte de DRAM.
- En modo Protegido (ya sea Flat o Segmentado) no hay restricciones respecto de la dirección de destino, dentro de los 4 Gbytes de capacidad de direccionamiento. Pero debe estar el sistema correctamente inicializado (tablas de descriptores de segmento mínimas y demás delicias de la vida).



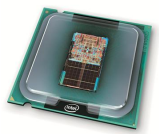
6.4. Salto a memoria DRAM

- Aquí es donde se puede producir un problema si no se hicieron bien las cosas.
- El programa debe haberse autocopiado de NVRAM a DRAM
- El salto se implementa mediante un salto FAR.
- En modo 16 bits o real hay que saltar a una dirección dentro del primer Mbyte de DRAM.
- En modo Protegido (ya sea Flat o Segmentado) no hay restricciones respecto de la dirección de destino, dentro de los 4 Gbytes de capacidad de direccionamiento. Pero debe estar el sistema correctamente inicializado (tablas de descriptores de segmento mínimas y demás delicias de la vida).



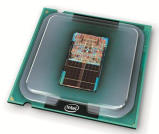
6.4. Salto a memoria DRAM

- Aquí es donde se puede producir un problema si no se hicieron bien las cosas.
- El programa debe haberse autocopiado de NVRAM a DRAM
- El salto se implementa mediante un salto FAR.
- En modo 16 bits o real hay que saltar a una dirección dentro del primer Mbyte de DRAM.
- En modo Protegido (ya sea Flat o Segmentado) no hay restricciones respecto de la dirección de destino, dentro de los 4 Gbytes de capacidad de direccionamiento. Pero debe estar el sistema correctamente inicializado (tablas de descriptores de segmento mínimas y demás delicias de la vida).



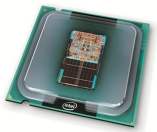
6.4. Salto a memoria DRAM

- Aquí es donde se puede producir un problema si no se hicieron bien las cosas.
- El programa debe haberse autocopiado de NVRAM a DRAM
- El salto se implementa mediante un salto FAR.
- En modo 16 bits o real hay que saltar a una dirección dentro del primer Mbyte de DRAM.
- En modo Protegido (ya sea Flat o Segmentado) no hay restricciones respecto de la dirección de destino, dentro de los 4 Gbytes de capacidad de direccionamiento. Pero debe estar el sistema correctamente inicializado (tablas de descriptores de segmento mínimas y demás delicias de la vida).



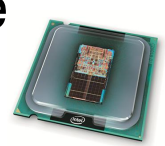
6.4. Salto a memoria DRAM

- Aquí es donde se puede producir un problema si no se hicieron bien las cosas.
- El programa debe haberse autocopiado de NVRAM a DRAM
- El salto se implementa mediante un salto FAR.
- En modo 16 bits o real hay que saltar a una dirección dentro del primer Mbyte de DRAM.
- En modo Protegido (ya sea Flat o Segmentado) no hay restricciones respecto de la dirección de destino, dentro de los 4 Gbytes de capacidad de direccionamiento. Pero debe estar el sistema correctamente inicializado (tablas de descriptores de segmento mínimas y demás delicias de la vida).



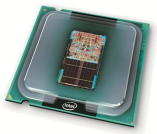
6.4. Salto a memoria DRAM

- Aquí es donde se puede producir un problema si no se hicieron bien las cosas.
- El programa debe haberse autocopiado de NVRAM a DRAM
- El salto se implementa mediante un salto FAR.
- En modo 16 bits o real hay que saltar a una dirección dentro del primer Mbyte de DRAM.
- En modo Protegido (ya sea Flat o Segmentado) no hay restricciones respecto de la dirección de destino, dentro de los 4 Gbytes de capacidad de direccionamiento. Pero debe estar el sistema correctamente inicializado (tablas de descriptores de segmento mínimas y demás delicias de la vida).



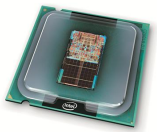
7. Habilitaciones de dispositivos misceláneos

- Este ítem es plataforma dependiente y se compone de un conjunto de dispositivos que surgen del análisis de lso esquemáticos de hardware.
- Los típicos dispositivos que están presentes en todos los chipsets (con las variantes de implementación de cada caso) son:
  1. Programación del chip de reloj.
  2. GPIO, igualmente debe consultarse el Chipset EDS Vol. 1 (NDA mediante) para tener los detalles.



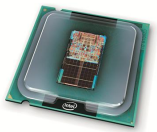
7. Habilitaciones de dispositivos misceláneos

- Este ítem es plataforma dependiente y se compone de un conjunto de dispositivos que surgen del análisis de lso esquemáticos de hardware.
- Los típicos dispositivos que están presentes en todos los chipsets (con las variantes de implementación de cada caso) son:
  1. Programación del chip de reloj
  2. GPIO, igualmente debe consultarse el Chipset EDS (Intel Confidential (NDA mediante) para tener los detalles



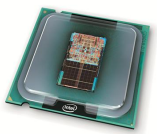
7. Habilitaciones de dispositivos misceláneos

- Este ítem es plataforma dependiente y se compone de un conjunto de dispositivos que surgen del análisis de lso esquemáticos de hardware.
- Los típicos dispositivos que están presentes en todos los chipsets (con las variantes de implementación de cada caso) son:
  1. Programación del chip de reloj
  2. GPIO. Igualmente debe consultarse el Chipset BIOS Writer Guide (NDA mediante) para tener los detalles.



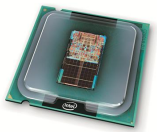
7. Habilitaciones de dispositivos misceláneos

- Este ítem es plataforma dependiente y se compone de un conjunto de dispositivos que surgen del análisis de lso esquemáticos de hardware.
- Los típicos dispositivos que están presentes en todos los chipsets (con las variantes de implementación de cada caso) son:
  1. Programación del chip de reloj
  2. GPIO. Igualmente debe consultarse el Chipset BIOS Writer Guide (NDA mediante) para tener los detalles.



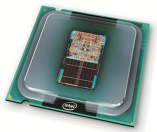
7. Habilitaciones de dispositivos misceláneos

- Este ítem es plataforma dependiente y se compone de un conjunto de dispositivos que surgen del análisis de Iso esquemáticos de hardware.
- Los típicos dispositivos que están presentes en todos los chipsets (con las variantes de implementación de cada caso) son:
  1. Programación del chip de reloj
  2. GPIO. Igualmente debe consultarse el Chipset BIOS Writer Guide (NDA mediante) para tener los detalles.



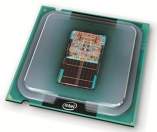
8. Habilitaciones de Interrupciones

- Los procesadores de intel pueden manejar las interrupciones mediante cualquiera de los siguientes métodos o combinación de ellos:
  1. Programmable Interrupt Controller (PIC)
  2. Local Advanced Programmable Interrupt Controller (APIC)
  3. Input/Output Advanced Programmable Interrupt Controller (IO×APIC)
  4. Messaged Signaled Interrupt (MSI)



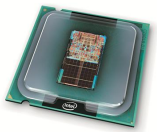
8. Habilitaciones de Interrupciones

- Los procesadores de intel pueden manejar las interrupciones mediante cualquiera de los siguientes métodos o combinación de ellos:
  1. Programmable Interrupt Controller (PIC)
  2. Local Advanced Programmable Interrupt Controller (APIC)
  3. Input /Output Advanced Programmable Interrupt Controller (IOxAPIC)
  4. Messaged Signaled Interrupt (MSI)



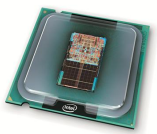
8. Habilitaciones de Interrupciones

- Los procesadores de intel pueden manejar las interrupciones mediante cualquiera de los siguientes métodos o combinación de ellos:
  1. Programmable Interrupt Controller (PIC)
  2. Local Advanced Programmable Interrupt Controller (APIC)
  3. Input /Output Advanced Programmable Interrupt Controller (IOxAPIC)
  4. Messaged Signaled Interrupt (MSI)



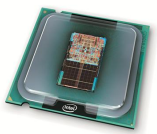
8. Habilitaciones de Interrupciones

- Los procesadores de intel pueden manejar las interrupciones mediante cualquiera de los siguientes métodos o combinación de ellos:
  1. Programmable Interrupt Controller (PIC)
  2. Local Advanced Programmable Interrupt Controller (APIC)
  3. Input /Output Advanced Programmable Interrupt Controller (IOxAPIC)
  4. Messaged Signaled Interrupt (MSI)



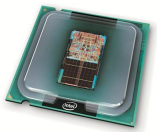
8. Habilitaciones de Interrupciones

- Los procesadores de intel pueden manejar las interrupciones mediante cualquiera de los siguientes métodos o combinación de ellos:
  1. Programmable Interrupt Controller (PIC)
  2. Local Advanced Programmable Interrupt Controller (APIC)
  3. Input /Output Advanced Programmable Interrupt Controller (IOxAPIC)
  4. Messaged Signaled Interrupt (MSI)

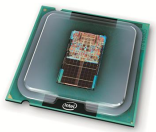


8. Habilitaciones de Interrupciones

- Los procesadores de intel pueden manejar las interrupciones mediante cualquiera de los siguientes métodos o combinación de ellos:
  1. Programmable Interrupt Controller (PIC)
  2. Local Advanced Programmable Interrupt Controller (APIC)
  3. Input /Output Advanced Programmable Interrupt Controller (IOxAPIC)
  4. Messaged Signaled Interrupt (MSI)

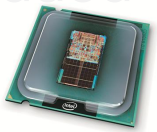


Es hora de cargar un Sistema Operativo



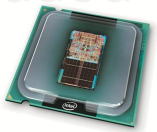
¿Quien dará los pasos siguientes?

- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



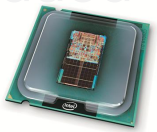
¿Quien dará los pasos siguientes?

- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



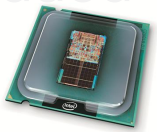
¿Quien dará los pasos siguientes?

- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



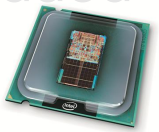
¿Quien dará los pasos siguientes?

- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



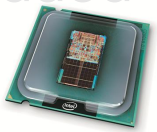
¿Quien dará los pasos siguientes?

- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



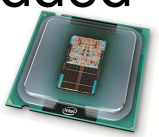
¿Quien dará los pasos siguientes?

- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



¿Quien dará los pasos siguientes?

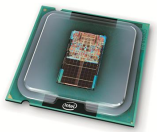
- La pregunta en esta instancia del desarrollo es: ¿Que recursos tenemos a la mano para usar?.
- Respuesta: Nuestro conocimiento del hardware de base, un compilador, un linker y alguna otra herramientas de desarrollo.
- Así comienza el desarrollo de cualquier computador.
- Incluso el de aquel simpático embedded que utilizaste hasta ahora.
- La pregunta que estás por hacerme es: ¿No están ya desarrollados?. ¿Para que necesitamos meternos en este problema?
- Mi respuesta es esta pregunta: ¿Te pusiste a pensar cual es la profesión del que diseñó la inicialización de tu PC o del embedded system que usaste en la carrera hasta ahora?. Pensá.....



Multiple choice. Necesario aprobar para seguir adelante

¿Cual es la skill de la gente que diseñó la inicialización de tu embedded?

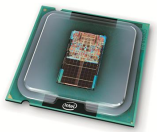
- **a.** Bombero
- **b.** Odontólogo
- **c.** Fisioterapeuta
- **d.** Chef
- **e.** Licenciado en Ciencias de la Computación
- **f.** Vendedor de Seguros
- **g.** Periodista
- **h.** Abogado
- **i.** Actor



Multiple choice. Necesario aprobar para seguir adelante

¿Cual es la skill de la gente que diseñó la inicialización de tu embedded?

- **a.** Bombero
- **b.** Odontólogo
- **c.** Fisioterapeuta
- **d.** Chef
- **e.** Licenciado en Ciencias de la Computación
- **f.** Vendedor de Seguros
- **g.** Periodista
- **h.** Abogado
- **i.** Actor



Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
  2. Cuando programamos el comportamiento es similar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
  2. Cuando programamos el comportamiento es similar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el icono llamado *Install*, *Setup*, o algo similar. Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.
     - Pidiendo recursos vía System Calls (malloc, fopen, free, printf, scanf, etc.). Esto es razonable.
     - Enviando requerimientos para acceder a la E/S. Esto también.
     - Usando librerías de código que nos facilitan la vida siempre que se pueda (no va a ser que usemos un código nuestro mas eficiente...).
     - En nombre de la productividad dejamos de pensar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.
     - Pidiendo recursos vía System Calls (*malloc*, *fopen*, *free*, *printf*, *scanf*, etc.). Esto es razonable.
     - Enviando requerimientos para acceder a la E/S. Esto también.
     - Usando librerías de código que nos facilitan la vida siempre que se pueda (no va a ser que usemos un código nuestro mas eficiente...).

     En nombre de la productividad dejamos de pensar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.
     - Pidiendo recursos vía System Calls (malloc, fopen, free, printf, scanf, etc.). Esto es razonable.
     - Enviando requerimientos para acceder a la E/S. Esto también.
     - Usando librerías de código que nos facilitan la vida siempre que se pueda (no va a ser que usemos un código nuestro mas eficiente...). En nombre de la productividad dejamos de pensar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.
     - Pidiendo recursos vía System Calls (malloc, fopen, free, printf, scanf, etc.). Esto es razonable.
     - Enviando requerimientos para acceder a la E/S. Esto también.
     - Usando librerías de código que nos facilitan la vida siempre que se pueda (no va a ser que usemos un código nuestro mas eficiente...). En nombre de la productividad dejamos de pensar.

Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.
     - Pidiendo recursos vía System Calls (malloc, fopen, free, printf, scanf, etc.). Esto es razonable.
     - Enviando requerimientos para acceder a la E/S. Esto también.
     - Usando librerías de código que nos facilitan la vida siempre que se pueda (no va a ser que usemos un código nuestro mas eficiente...). En nombre de la productividad dejamos de pensar.

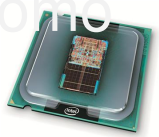
Solo si aprobaste el Multiple choice.

La cuestión es dejar de pensar y actuar como usuarios

- **Diagnóstico**: Estamos acostumbrados a usar nuestra PC, aun para trabajar con un embebido...
- Esto nos induce a pensar como usuarios finales **siempre**.
  1. Cuando usamos aplicaciones en nuestra PC.
     - Instalación: ¿Alguna vez te compilaste una aplicación a partir de sus fuentes?.¿¡Compilar!? ¿Para que?. Gracias Wizards!!.
     - Buscamos el ícono llamado *Install*, *Setup*, o algo similar, Doble click al encontrarlo, y Botón Aceptar hasta las últimas consecuencias...
     - ¿Y cuando tenemos un problema?... botón de reset... :(
  2. Cuando programamos el comportamiento es similar.
     - Pidiendo recursos vía System Calls (malloc, fopen, free, printf, scanf, etc.). Esto es razonable.
     - Enviando requerimientos para acceder a la E/S. Esto también.
     - Usando librerías de código que nos facilitan la vida siempre que se pueda (no va a ser que usemos un código nuestro mas eficiente...). En nombre de la productividad dejamos de pensar.

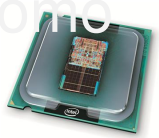
Solo para profesionales de la computación

- Lo dicho en el slide anterior no es válido para usuarios generales, como por ejemplo el resto de las opciones del múltiple choice anterior. Mas aún. Para este público el comportamiento descripto es el esperable.
- Pero Uds. son **otro público**, básicamente porque **eligieron serlo**. Por eso están aquí. ¿no?
- Así que a enterarse: Tu trabajo es y será siempre, **entender** como funcionan los sistemas electrónicos que te toque enfrentar, para poderlos diseñar, mejorar, o corregir. En este caso un computador, pero lo mismo vale para un transmisor de RF, o un sistema de control de lazo cerrado.
- Creeme: Entender cuesta. Pero hace la diferencia. Implica profundizar hasta dominar la tecnología. Algo que pesando como usuario no vas a conseguir...



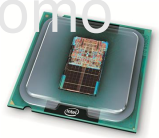
Solo para profesionales de la computación

- Lo dicho en el slide anterior no es válido para usuarios generales, como por ejemplo el resto de las opciones del múltiple choice anterior. Mas aún. Para este público el comportamiento descripto es el esperable.
- Pero Uds. son **otro público**, básicamente porque **eligieron serlo**. Por eso están aquí. ¿no?
- Así que a enterarse: Tu trabajo es y será siempre, **entender** como funcionan los sistemas electrónicos que te toque enfrentar, para poderlos diseñar, mejorar, o corregir. En este caso un computador, pero lo mismo vale para un transmisor de RF, o un sistema de control de lazo cerrado.
- Creeme: Entender cuesta. Pero hace la diferencia. Implica profundizar hasta dominar la tecnología. Algo que pesando como usuario no vas a conseguir...



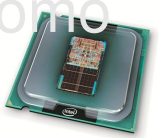
Solo para profesionales de la computación

- Lo dicho en el slide anterior no es válido para usuarios generales, como por ejemplo el resto de las opciones del múltiple choice anterior. Mas aún. Para este público el comportamiento descripto es el esperable.
- Pero Uds. son **otro público**, básicamente porque **eligieron serlo**. Por eso están aquí. ¿no?
- Así que a enterarse: Tu trabajo es y será siempre, **entender** como funcionan los sistemas electrónicos que te toque enfrentar, para poderlos diseñar, mejorar, o corregir. En este caso un computador, pero lo mismo vale para un transmisor de RF, o un sistema de control de lazo cerrado.
- Creeme: Entender cuesta. Pero hace la diferencia. Implica profundizar hasta dominar la tecnología. Algo que pesando como usuario no vas a conseguir...



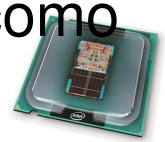
Solo para profesionales de la computación

- Lo dicho en el slide anterior no es válido para usuarios generales, como por ejemplo el resto de las opciones del múltiple choice anterior. Mas aún. Para este público el comportamiento descripto es el esperable.
- Pero Uds. son **otro público**, básicamente porque **eligieron serlo**. Por eso están aquí. ¿no?
- Así que a enterarse: Tu trabajo es y será siempre, **entender** como funcionan los sistemas electrónicos que te toque enfrentar, para poderlos diseñar, mejorar, o corregir. En este caso un computador, pero lo mismo vale para un transmisor de RF, o un sistema de control de lazo cerrado.
- Creeme: Entender cuesta. Pero hace la diferencia. Implica profundizar hasta dominar la tecnología. Algo que pesando como usuario no vas a conseguir...

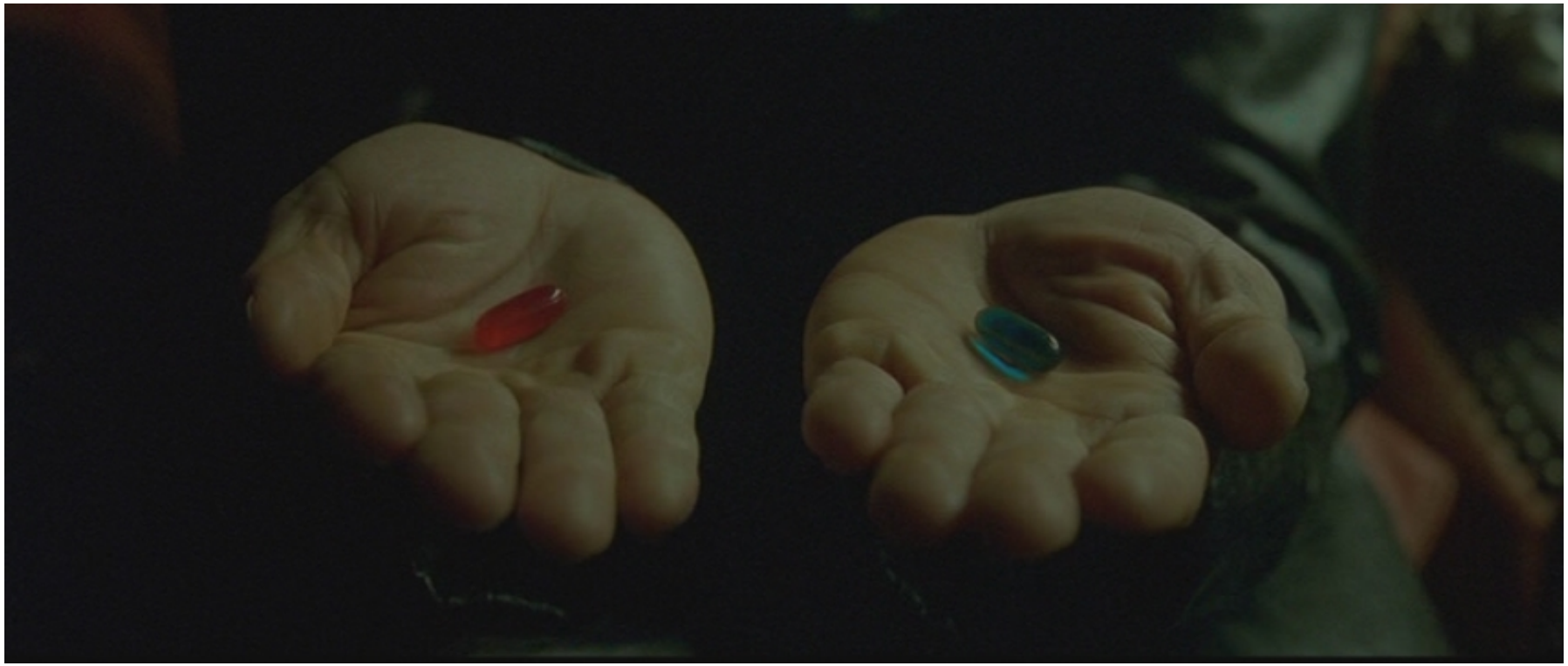


Solo para profesionales de la computación

- Lo dicho en el slide anterior no es válido para usuarios generales, como por ejemplo el resto de las opciones del múltiple choice anterior. Mas aún. Para este público el comportamiento descripto es el esperable.
- Pero Uds. son **otro público**, básicamente porque **eligieron serlo**. Por eso están aquí. ¿no?
- Así que a enterarse: Tu trabajo es y será siempre, **entender** como funcionan los sistemas electrónicos que te toque enfrentar, para poderlos diseñar, mejorar, o corregir. En este caso un computador, pero lo mismo vale para un transmisor de RF, o un sistema de control de lazo cerrado.
- Creeme: Entender cuesta. Pero hace la diferencia. Implica profundizar hasta dominar la tecnología. Algo que pesando como usuario no vas a conseguir...



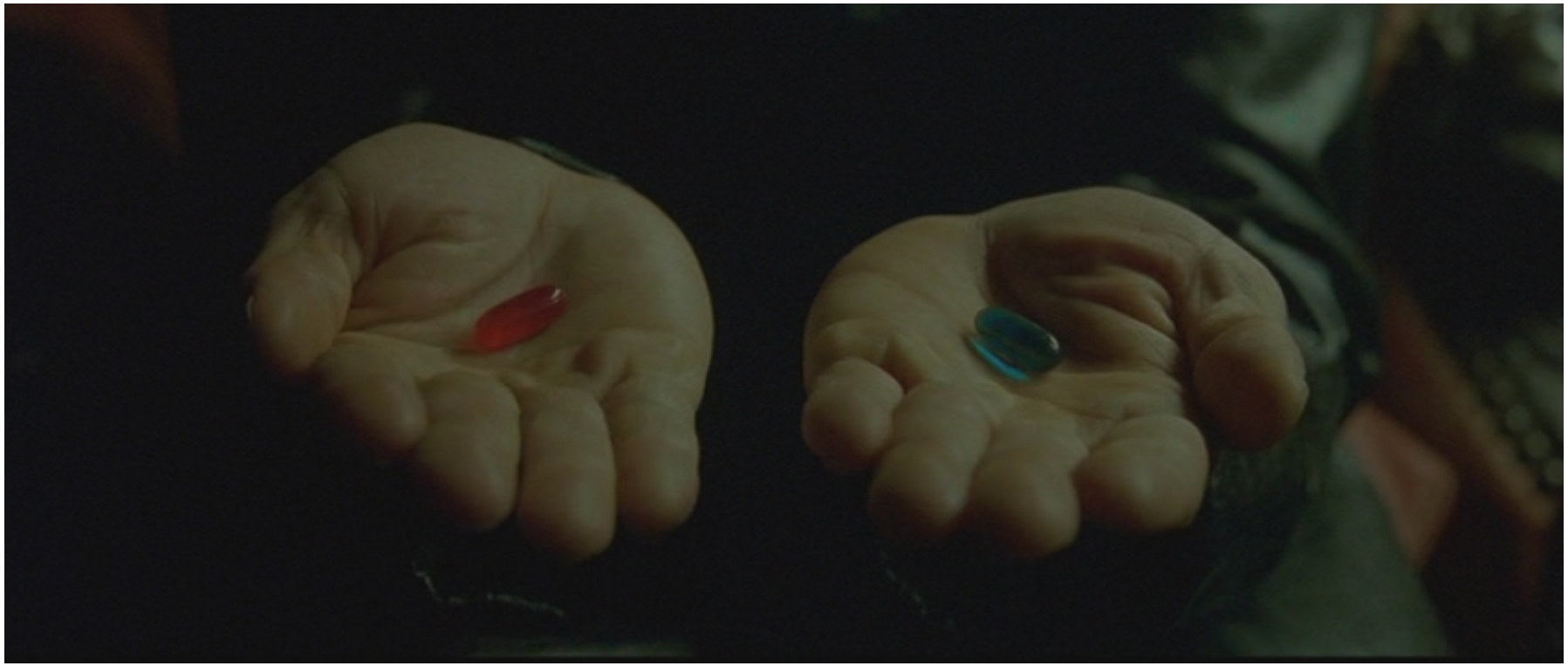
Es momento de revalidar la elección



The choice

O nos quedamos con el wizard que nos resuelve la vida sin tener que pensar... O nos decidimos a enfrentar las cosas como son realmente, y entenderlas, aprendiendo, si es necesario, a hacer todo desde cero y a pulmón.

Es momento de revalidar la elección



The choice

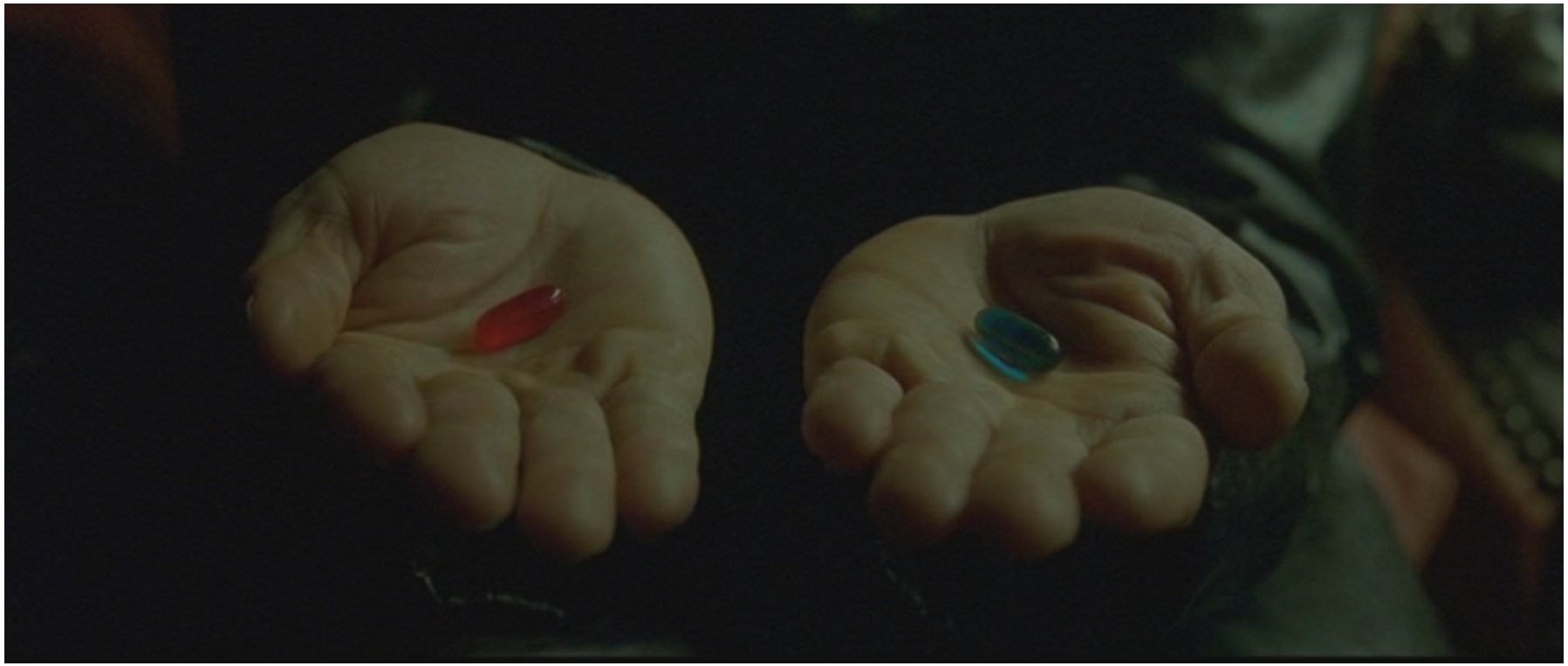
O nos quedamos con el wizard que nos resuelve la vida sin tener que pensar... O nos decidimos a enfrentar las cosas como son realmente, y entenderlas, aprendiendo, si es necesario, a hacer todo desde cero y a pulmón.

Retomando nuestro problema

Arrancamos el procesador y estamos en modo real

Para iniciar su operación en modo real no se requiere mas que inicializar un sistema de interrupciones con las reglas de modo real (para un 8086 diseñado en 1978), con los vectores de interrupción apuntando a código diseñado para operar en Modo Real, y tener disponibles las correspondientes funciones para cada handler de interrupción.
Si nos pensamos quedar en este pequeño microclima de confort (desperdiciando el 99,999 % de los recursos del procesador :-/), se necesita un mínimo kernel que administre la ejecución de los programas que compongan en rango de aplicaciones, ejecutándolas una a la vez (leíste bien... Primero una , y recién cuando finaliza, podés ejecutar otra aplicación).

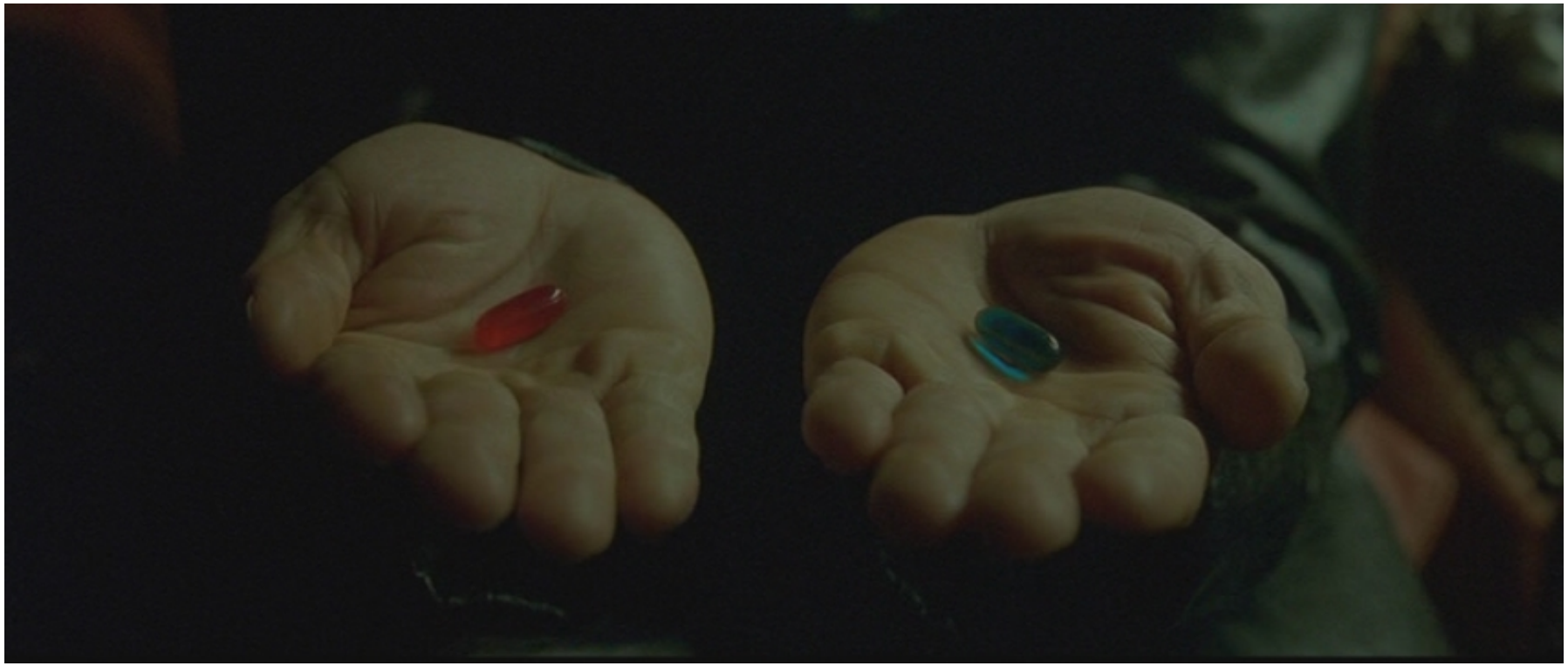
Otra vez...



The choice

La píldora azul nos deja en este nanoclima confortable. Fin de la presentación.
La píldora roja nos lleva al mundo real, donde utilizaremos TODOS los recursos del procesador para construir un Sistema Operativo...aunque no nos guste lo que vamos a encontrar.

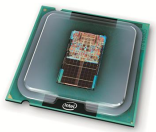
Otra vez...



The choice

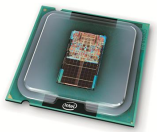
La píldora azul nos deja en este nanoclima confortable. Fin de la presentación.
La píldora roja nos lleva al mundo real, donde utilizaremos TODOS los recursos del procesador para construir un Sistema Operativo...aunque no nos guste lo que vamos a encontrar.

FAQ's



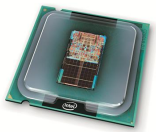
Q: ¿Vamos a desarrollar un sistema operativo???

A: La construcción de un Sistema Operativo es a veces una tarea gigantesca, como por ejemplo Linux, pero en ocasiones puede requerir un número mucho menor de rutinas que, aunque de muy bajo nivel, provean un conjunto de recursos base suficientes para administrar un sistema de menor tamaño, como un embeeded system.



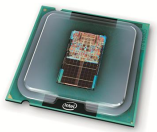
Q: ¿Vamos a desarrollar un sistema operativo???

A: La construcción de un Sistema Operativo es a veces una tarea gigantesca, como por ejemplo Linux, pero en ocasiones puede requerir un número mucho menor de rutinas que, aunque de muy bajo nivel, provean un conjunto de recursos base suficientes para administrar un sistema de menor tamaño, como un embeeded system.



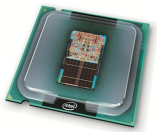
Q: ¿Para que?¿O acaso un embedded system no puede hostear Linux?

A: Por supuesto. Abundan implementaciones de Linux para Embedded Systems. Las embedded PC basadas en procesadores Atom, tambien pueden aceptar un Linux cualquiera. Aunque no todos los sistemas tienen un procesador con recursos de hardware suficiente para soportar Linux (por ahora... en cinco años hablamos otra vez).



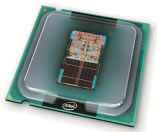
Q: ¿Para que?¿O acaso un embedded system no puede hostear Linux?

A: Por supuesto. Abundan implementaciones de Linux para Embedded Systems. Las embedded PC basadas en procesadores Atom, tambien pueden aceptar un Linux cualquiera. Aunque no todos los sistemas tienen un procesador con recursos de hardware suficiente para soportar Linux (por ahora... en cinco años hablamos otra vez).



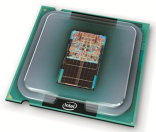
Q: ¿Para que necesitamos saber esto entonces? ¿No está todo hecho?

A: ¿Otra vez pensando como usuario? Además si un tal Linus Torvalds se hubiese quedado en esta pregunta hoy solo existiría Windows como alternativa para nuestra PC. (Vade retro!)

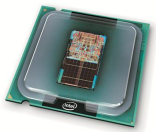


Q: ¿Para que necesitamos saber esto entonces? ¿No está todo hecho?

A: ¿Otra vez pensando como usuario? Además si un tal Linus Torvalds se hubiese quedado en esta pregunta hoy solo existiría Windows como alternativa para nuestra PC. (Vade retro!)

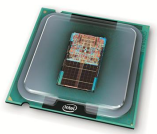


Ingresando a modo protegido



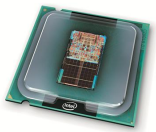
Requerimientos de los Sistemas Operativos Multitasking

- Área de memoria exclusiva para cada tarea para almacenar su código y sus datos. (Área Local).
- Área de memoria común a todas las aplicaciones, para que éstas puedan acceder a datos globales del sistema, o a código propio del Sistema Operativo de modo de permitir la comunicación entre las aplicaciones. (Área Global).
- Cada tarea podrá acceder únicamente a su Área Local y al Área Global, pero nunca podrá acceder al Área Local de otra tarea. De este modo el Sistema Operativo garantiza la integridad (PROTECCION ;)) del código y de los datos propios de cada tarea.
- Alta velocidad de procesamiento
- Gran capacidad de Direccionamiento de memoria



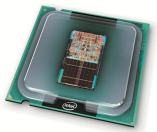
Requerimientos de los Sistemas Operativos Multitasking

- Amplio espacio de direccionamiento para memoria RAM
- Capacidad de Gestión de memoria de cada tarea por el método de Memoria Virtual
- Capacidad de implementar Multitarea de manera rápida y segura.
- En cada momento la CPU ejecuta una tarea de la lista que mantiene, poniendo a su disposición todos los recursos de hardware de la máquina, incluyendo la cantidad de memoria requerida por la aplicación.



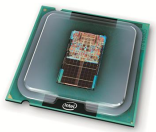
Finalmente... ¿Que es Modo Protegido?

- Es el conjunto de recursos de hardware (Dije Hardware. O sea Electrónica.) y sus reglas de funcionamiento que se requieren para darle sustento a un sistema operativo multitasking satisfaciendo los requerimientos anteriores.
- Su dominio permite entender como funcionan las cosas en un sistema real que puede ser un súper servidor, o un embedded.Y en definitiva es nuestro trabajo.
- Es decir,... es tomar la píldora roja.



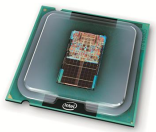
¿Como encarar la tarea?

Dejando de pensar como un programador de aplicaciones, y comenzando a pensar como el programador de un sistema operativo.



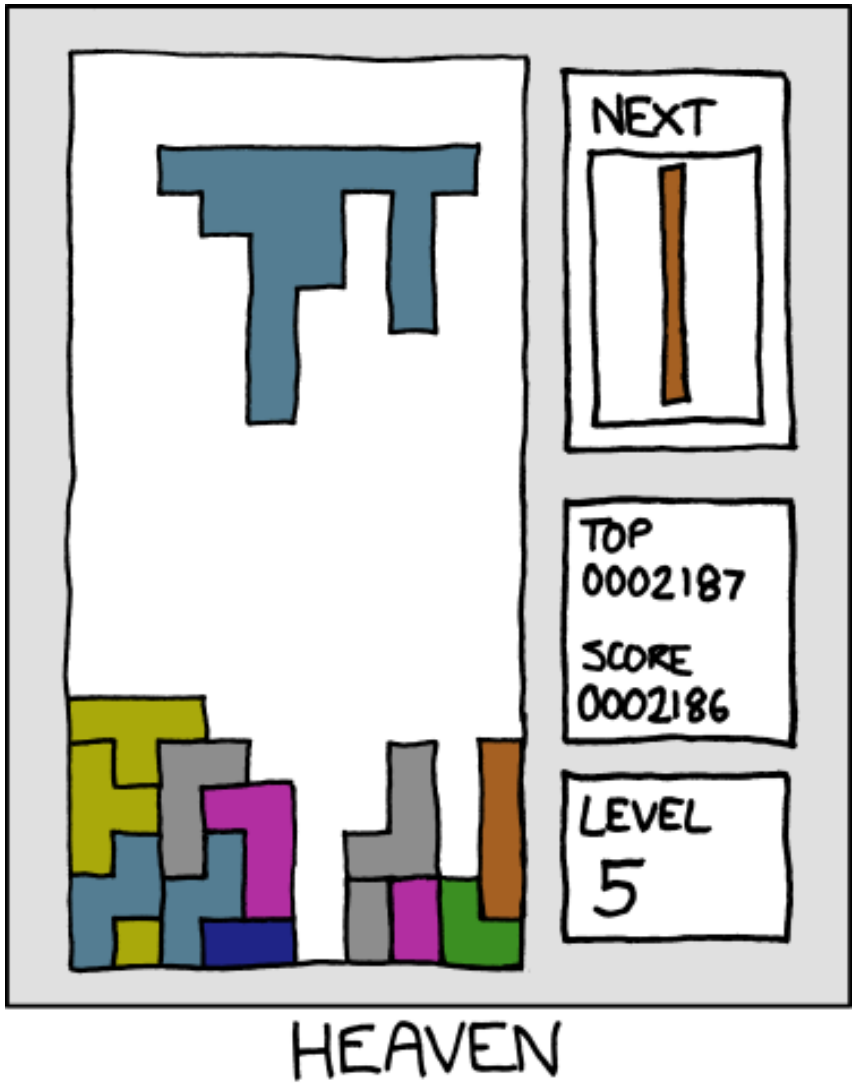
¿Como encarar la tarea?

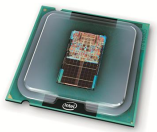
Dejando de pensar como un programador de aplicaciones, y comenzando a pensar como el programador de un sistema operativo.



No vamos a mentir. No es fácil

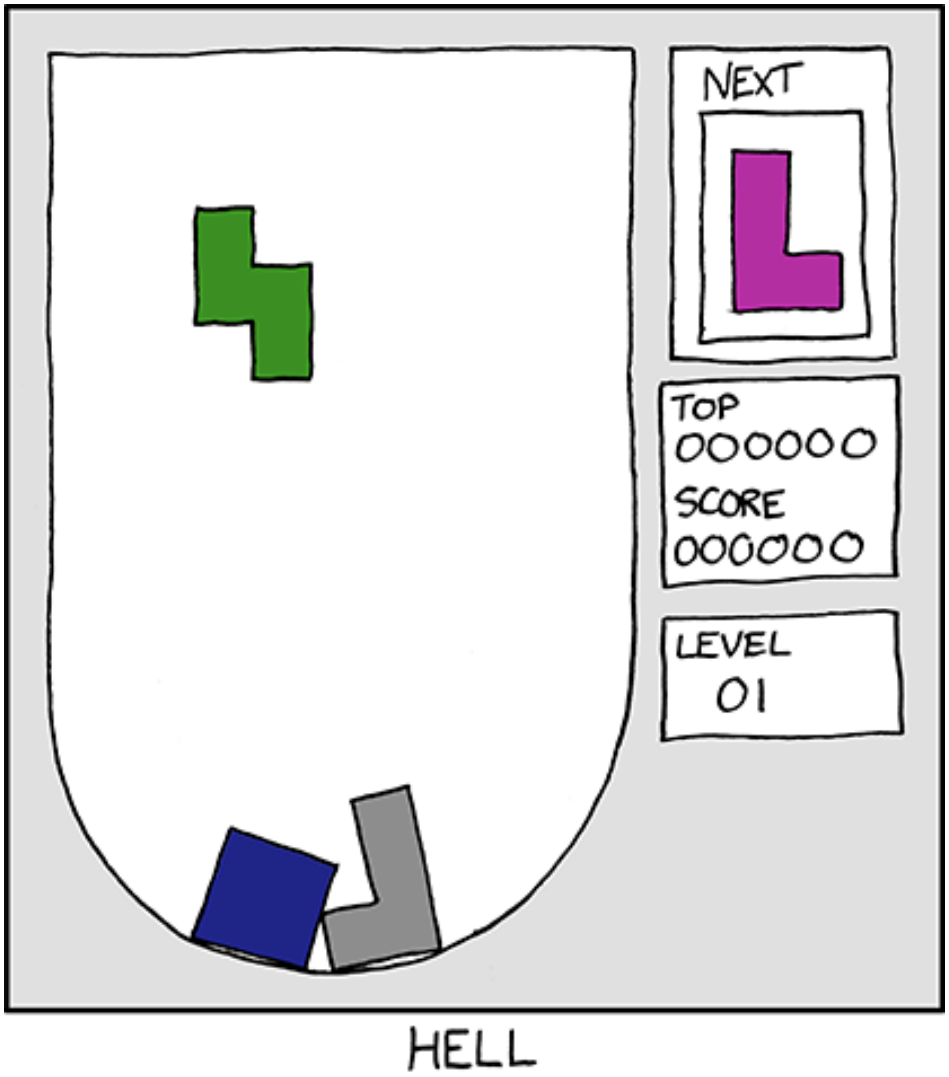
Programación a nivel de aplicación...

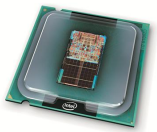




En ocasiones vas a experimentar algo como...

Programación en Modo Protegido desde cero...



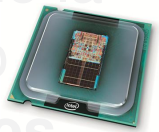


“Wellcome... To the real world”

¡Importante!

Para pasar a Modo Protegido, es suficiente con setear el bit ***CRO.PE***. Pero antes de esta simple e inocente acción, es necesario preparar un entorno mínimo para que todo funcione adecuadamente, y no se estrelle el nano kernel que nos proponemos desarrollar.

1. Programar una ***IDT*** con los descriptores de Excepciones necesarios para operar el sistema, cada uno referenciado a un mínimo handler que al menos sirva para detener el procesador en caso de generarse alguna excepción, y con los descriptores de Interrupción apuntados a los handlers de interrupción del hardware que necesitamos utilizar.
2. Programar una ***GDT*** con los descriptores de Código y Datos que necesitemos mínimamente para iniciar el sistema.
3. Inicializar el registro ***GDTR***, con la dirección base y tamaño de la tabla respectiva. ***IDTR*** puede inicializarse aquí, o ya en modo protegido pero **siempre antes de habilitar las interrupciones**.

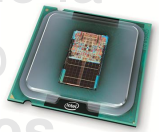


"Wellcome... To the real world"

¡Importante!

Para pasar a Modo Protegido, es suficiente con setear el bit ***CRO.PE***. Pero antes de esta simple e inocente acción, es necesario preparar un entorno mínimo para que todo funcione adecuadamente, y no se estrelle el nano kernel que nos proponemos desarrollar.

1. Programar una ***IDT*** con los descriptores de Excepciones necesarios para operar el sistema, cada uno referenciado a un mínimo handler que al menos sirva para detener el procesador en caso de generarse alguna excepción, y con los descriptores de Interrupción apuntados a los handlers de interrupción del hardware que necesitamos utilizar.
2. Programar una ***GDT*** con los descriptores de Código y Datos que necesitemos mínimamente para iniciar el sistema.
3. Inicializar el registro ***GDTR***, con la dirección base y tamaño de la tabla respectiva. ***IDTR*** puede inicializarse aquí, o ya en modo protegido pero **siempre antes de habilitar las interrupciones**.

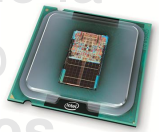


"Wellcome... To the real world"

¡Importante!

Para pasar a Modo Protegido, es suficiente con setear el bit ***CRO.PE***. Pero antes de esta simple e inocente acción, es necesario preparar un entorno mínimo para que todo funcione adecuadamente, y no se estrelle el nano kernel que nos proponemos desarrollar.

1. Programar una ***IDT*** con los descriptores de Excepciones necesarios para operar el sistema, cada uno referenciado a un mínimo handler que al menos sirva para detener el procesador en caso de generarse alguna excepción, y con los descriptores de Interrupción apuntados a los handlers de interrupción del hardware que necesitamos utilizar.
2. Programar una ***GDT*** con los descriptores de Código y Datos que necesitemos mínimamente para iniciar el sistema.
3. Inicializar el registro ***GDTR***, con la dirección base y tamaño de la tabla respectiva. ***IDTR*** puede inicializarse aquí, o ya en modo protegido pero **siempre antes de habilitar las interrupciones**.

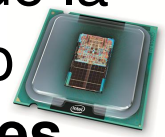


"Wellcome... To the real world"

¡Importante!

Para pasar a Modo Protegido, es suficiente con setear el bit ***CRO.PE***. Pero antes de esta simple e inocente acción, es necesario preparar un entorno mínimo para que todo funcione adecuadamente, y no se estrelle el nano kernel que nos proponemos desarrollar.

1. Programar una ***IDT*** con los descriptores de Excepciones necesarios para operar el sistema, cada uno referenciado a un mínimo handler que al menos sirva para detener el procesador en caso de generarse alguna excepción, y con los descriptores de Interrupción apuntados a los handlers de interrupción del hardware que necesitamos utilizar.
2. Programar una ***GDT*** con los descriptores de Código y Datos que necesitemos mínimamente para iniciar el sistema.
3. Inicializar el registro ***GDTR***, con la dirección base y tamaño de la tabla respectiva. ***IDTR*** puede inicializarse aquí, o ya en modo protegido pero **siempre antes de habilitar las interrupciones**.



“Wellcome... To the real world”

4. Armar al menos una ***TSS***.
5. Armar una ***LDT*** (opcional... solo para valientes :–))
6. Si vamos a administrar la memoria por paginación, entonces hay que armar al menos un Directorio de Tabla de Páginas, con al menos una entrada válida que referencia a una Tabla de Páginas, que a su vez debe estar correctamente inicializada con las referencias a las áreas de memoria que vamos a utilizar al inicio de la operación del sistema.
7. Un segmento de Código que contenga el código que vamos a ejecutar ni bien se ponga al procesador en Modo Protegido, y los handlers de interrupción y de excepción. Este segmento debe tener un descriptor debidamente inicializado en la ***GDT***.
8. Inicializar los registros de Control del procesador. ***CR3*** con la dirección física de inicio del Directorio de Tablas de Páginas, y en ***CR4***, setear algún modo de paginación larga en caso de querer utilizarse (Hacerlo en modo protegido con la Paginación habilitada genera una excepción #GP).

“Wellcome... To the real world”

4. Armar al menos una ***TSS***.
5. Armar una ***LDT*** (opcional... solo para valientes :–))
6. Si vamos a administrar la memoria por paginación, entonces hay que armar al menos un Directorio de Tabla de Páginas, con al menos una entrada válida que referencia a una Tabla de Páginas, que a su vez debe estar correctamente inicializada con las referencias a las áreas de memoria que vamos a utilizar al inicio de la operación del sistema.
7. Un segmento de Código que contenga el código que vamos a ejecutar ni bien se ponga al procesador en Modo Protegido, y los handlers de interrupción y de excepción. Este segmento debe tener un descriptor debidamente inicializado en la ***GDT***.
8. Inicializar los registros de Control del procesador. ***CR3*** con la dirección física de inicio del Directorio de Tablas de Páginas, y en ***CR4***, setear algún modo de paginación larga en caso de querer utilizarse (Hacerlo en modo protegido con la Paginación habilitada genera una excepción #GP).

“Wellcome... To the real world”

4. Armar al menos una ***TSS***.
5. Armar una ***LDT*** (opcional... solo para valientes :–))
6. Si vamos a administrar la memoria por paginación, entonces hay que armar al menos un Directorio de Tabla de Páginas, con al menos una entrada válida que referencia a una Tabla de Páginas, que a su vez debe estar correctamente inicializada con las referencias a las áreas de memoria que vamos a utilizar al inicio de la operación del sistema.
7. Un segmento de Código que contenga el código que vamos a ejecutar ni bien se ponga al procesador en Modo Protegido, y los handlers de interrupción y de excepción. Este segmento debe tener un descriptor debidamente inicializado en la ***GDT***.
8. Inicializar los registros de Control del procesador. ***CR3*** con la dirección física de inicio del Directorio de Tablas de Páginas, y en ***CR4***, setear algún modo de paginación larga en caso de querer utilizarse (Hacerlo en modo protegido con la Paginación habilitada genera una excepción #GP).

“Wellcome... To the real world”

4. Armar al menos una ***TSS***.
5. Armar una ***LDT*** (opcional... solo para valientes :–))
6. Si vamos a administrar la memoria por paginación, entonces hay que armar al menos un Directorio de Tabla de Páginas, con al menos una entrada válida que referencia a una Tabla de Páginas, que a su vez debe estar correctamente inicializada con las referencias a las áreas de memoria que vamos a utilizar al inicio de la operación del sistema.
7. Un segmento de Código que contenga el código que vamos a ejecutar ni bien se ponga al procesador en Modo Protegido, y los handlers de interrupción y de excepción. Este segmento debe tener un descriptor debidamente inicializado en la ***GDT***.
8. Inicializar los registros de Control del procesador. ***CR3*** con la dirección física de inicio del Directorio de Tablas de Páginas, y en ***CR4***, setear algún modo de paginación larga en caso de querer utilizarse (Hacerlo en modo protegido con la Paginación habilitada genera una excepción #GP).

“Wellcome... To the real world”

4. Armar al menos una ***TSS***.
5. Armar una ***LDT*** (opcional... solo para valientes :–))
6. Si vamos a administrar la memoria por paginación, entonces hay que armar al menos un Directorio de Tabla de Páginas, con al menos una entrada válida que referencia a una Tabla de Páginas, que a su vez debe estar correctamente inicializada con las referencias a las áreas de memoria que vamos a utilizar al inicio de la operación del sistema.
7. Un segmento de Código que contenga el código que vamos a ejecutar ni bien se ponga al procesador en Modo Protegido, y los handlers de interrupción y de excepción. Este segmento debe tener un descriptor debidamente inicializado en la ***GDT***.
8. Inicializar los registros de Control del procesador. ***CR3*** con la dirección física de inicio del Directorio de Tablas de Páginas, y en ***CR4***, setear algún modo de paginación larga en caso de querer utilizarse (Hacerlo en modo protegido con la Paginación habilitada genera una excepción #GP).

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

¿Y ahora?

Hay que armar paulatinamente un kernel

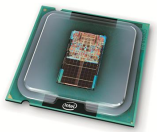
- Las definiciones que se tomen dependen de cada proyecto.
- Se debe definir un scheduler que se invoque periódicamente (desde el timer tick típicamente), capaz de de conmutar las diferentes tareas, asignándoles prioridades.
- Diseñar un sistema de protección que asegure que cada tarea trabajando en Modo User, tenga un área de memoria exclusiva inaccesible por el resto de las tareas del sistema, y acceso a los servicios del kernel ubicados en el máximo nivel de privilegio.
- Un sistema de Device Drivers, capaz de manejar una consola, un disco, un mouse, un dispositivo de comunicación serie, etc.
- Un file system para alojar las tareas en forma de archivos.
- Un loader para acceder al disco y cargar las tareas en memoria para su ejecución.
- etc. etc

Para trabajar en 64 bits hay que estar en Modo Protegido

¡Importante!

Para pasar a modo IA-32e es necesario partir de Modo Protegido, y tener previamente el bit ***CR4.PAE*** activo para (modo largo).
Asegurarse que todo el código que realizará la operación descripta a continuación resida en una página con identity mapping.

1. Si el procesador está en modo protegido con paginación habilitada y no está habilitado Physical Address Extension (PAE), antes que nada poner ***CR0.PG*** en 0 para deshabilitar paginación.
2. Habilitar Physical Address Extension, poniendo ***CR4.PAE*** = 1.
3. Inicializar ***CR3*** con la dirección física en que inicia el Page Map Directory Level 4 (PLM4).
4. Habilitar el Modo IA-32e, seteando el bit LME del Model Specific Register ***IA32_EFER***.
5. Habilitar paginación con ***CR0.PG*** = 1. Esto provocará que el procesador setee además el bit ***IA32_EFER.LMA***.

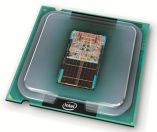


Para trabajar en 64 bits hay que estar en Modo Protegido

¡Importante!

Para pasar a modo IA-32e es necesario partir de Modo Protegido, y tener previamente el bit ***CR4.PAE*** activo para (modo largo).
Asegurarse que todo el código que realizará la operación descripta a continuación resida en una página con identity mapping.

1. Si el procesador está en modo protegido con paginación habilitada y no está habilitado Physical Address Extension (PAE), antes que nada poner ***CR0.PG*** en 0 para deshabilitar paginación.
2. Habilitar Physical Address Extension, poniendo ***CR4.PAE*** = 1.
3. Inicializar ***CR3*** con la dirección física en que inicia el Page Map Directory Level 4 (PLM4).
4. Habilitar el Modo IA-32e, seteando el bit LME del Model Specific Register ***IA32_EFER***.
5. Habilitar paginación con ***CR0.PG*** = 1. Esto provocará que el procesador setee además el bit ***IA32_EFER.LMA***.

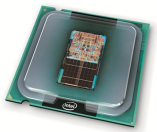


Para trabajar en 64 bits hay que estar en Modo Protegido

¡Importante!

Para pasar a modo IA-32e es necesario partir de Modo Protegido, y tener previamente el bit ***CR4.PAE*** activo para (modo largo).
Asegurarse que todo el código que realizará la operación descripta a continuación resida en una página con identity mapping.

1. Si el procesador está en modo protegido con paginación habilitada y no está habilitado Physical Address Extension (PAE), antes que nada poner ***CR0.PG*** en 0 para deshabilitar paginación.
2. Habilitar Physical Address Extension, poniendo ***CR4.PAE*** = 1.
3. Inicializar ***CR3*** con la dirección física en que inicia el Page Map Directory Level 4 (PLM4).
4. Habilitar el Modo IA-32e, seteando el bit LME del Model Specific Register ***IA32_EFER***.
5. Habilitar paginación con ***CR0.PG*** = 1. Esto provocará que el procesador setee además el bit ***IA32_EFER.LMA***.

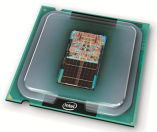


Para trabajar en 64 bits hay que estar en Modo Protegido

¡Importante!

Para pasar a modo IA-32e es necesario partir de Modo Protegido, y tener previamente el bit ***CR4.PAE*** activo para (modo largo).
Asegurarse que todo el código que realizará la operación descripta a continuación resida en una página con identity mapping.

1. Si el procesador está en modo protegido con paginación habilitada y no está habilitado Physical Address Extension (PAE), antes que nada poner ***CR0.PG*** en 0 para deshabilitar paginación.
2. Habilitar Physical Address Extension, poniendo ***CR4.PAE*** = 1.
3. Inicializar ***CR3*** con la dirección física en que inicia el Page Map Directory Level 4 (PLM4).
4. Habilitar el Modo IA-32e, seteando el bit LME del Model Specific Register ***IA32_EFER***.
5. Habilitar paginación con ***CR0.PG*** = 1. Esto provocará que el procesador setee además el bit ***IA32_EFER.LMA***.

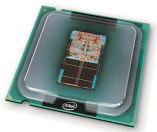


Para trabajar en 64 bits hay que estar en Modo Protegido

¡Importante!

Para pasar a modo IA-32e es necesario partir de Modo Protegido, y tener previamente el bit ***CR4.PAE*** activo para (modo largo).
Asegurarse que todo el código que realizará la operación descripta a continuación resida en una página con identity mapping.

1. Si el procesador está en modo protegido con paginación habilitada y no está habilitado Physical Address Extension (PAE), antes que nada poner ***CR0.PG*** en 0 para deshabilitar paginación.
2. Habilitar Physical Address Extension, poniendo ***CR4.PAE*** = 1.
3. Inicializar ***CR3*** con la dirección física en que inicia el Page Map Directory Level 4 (PLM4).
4. Habilitar el Modo IA-32e, seteando el bit LME del Model Specific Register ***IA32_EFER***.
5. Habilitar paginación con ***CR0.PG*** = 1. Esto provocará que el procesador setee además el bit ***IA32_EFER.LMA***.

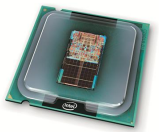


Para trabajar en 64 bits hay que estar en Modo Protegido

¡Importante!

Para pasar a modo IA-32e es necesario partir de Modo Protegido, y tener previamente el bit ***CR4.PAE*** activo para (modo largo).
Asegurarse que todo el código que realizará la operación descripta a continuación resida en una página con identity mapping.

1. Si el procesador está en modo protegido con paginación habilitada y no está habilitado Physical Address Extension (PAE), antes que nada poner ***CR0.PG*** en 0 para deshabilitar paginación.
2. Habilitar Physical Address Extension, poniendo ***CR4.PAE*** = 1.
3. Inicializar ***CR3*** con la dirección física en que inicia el Page Map Directory Level 4 (PLM4).
4. Habilitar el Modo IA-32e, seteando el bit LME del Model Specific Register ***IA32_EFER***.
5. Habilitar paginación con ***CR0.PG*** = 1. Esto provocará que el procesador setee además el bit ***IA32_EFER.LMA***.



¿Nada mas?

Las Tablas de Paginación deben residir en los primeros 4 Gbytes de memoria antes de habilitar el modo IA-32e.

Condiciones

El procesador chequea los bits ***CR0.PG***, ***CR4.PAE***, y ***IA32_EFER.LME*** para detectar inconsistencias. En cualquiera de los casos siguientes genera una excepción #GP.

- Si se está en modo protegido y se intenta entrar o salir del modo IA-32e con ***CR0.PG*** = 1
- Si en modo IA-32e se activa PAE antes de activar Paginación.
- Si se desactiva PAE en Modo IA-32e
- Si se intenta activar el modo IA-32e desde un código ubicado en un segmento de código de 64 bits.
- Si el registro TR tiene cargada una TSS de 16 bits.

¿Nada mas?

Las Tablas de Paginación deben residir en los primeros 4 Gbytes de memoria antes de habilitar el modo IA-32e.

Condiciones

El procesador chequea los bits ***CR0.PG***, ***CR4.PAE***, y ***IA32_EFER.LME*** para detectar inconsistencias. En cualquiera de los casos siguientes genera una excepción #GP.

- Si se está en modo protegido y se intenta entrar o salir del modo IA-32e con ***CR0.PG*** = 1
- Si en modo IA-32e se activa PAE antes de activar Paginación.
- Si se desactiva PAE en Modo IA-32e
- Si se intenta activar el modo IA-32e desde un código ubicado en un segmento de código de 64 bits.
- Si el registro TR tiene cargada una TSS de 16 bits.

¿Nada mas?

Las Tablas de Paginación deben residir en los primeros 4 Gbytes de memoria antes de habilitar el modo IA-32e.

Condiciones

El procesador chequea los bits ***CR0.PG***, ***CR4.PAE***, y ***IA32_EFER.LME*** para detectar inconsistencias. En cualquiera de los casos siguientes genera una excepción #GP.

- Si se está en modo protegido y se intenta entrar o salir del modo IA-32e con ***CR0.PG*** = 1
- Si en modo IA-32e se activa PAE antes de activar Paginación.
- Si se desactiva PAE en Modo IA-32e
- Si se intenta activar el modo IA-32e desde un código ubicado en un segmento de código de 64 bits.
- Si el registro TR tiene cargada una TSS de 16 bits.

¿Nada mas?

Las Tablas de Paginación deben residir en los primeros 4 Gbytes de memoria antes de habilitar el modo IA-32e.

Condiciones

El procesador chequea los bits ***CR0.PG***, ***CR4.PAE***, y ***IA32_EFER.LME*** para detectar inconsistencias. En cualquiera de los casos siguientes genera una excepción #GP.

- Si se está en modo protegido y se intenta entrar o salir del modo IA-32e con ***CR0.PG*** = 1
- Si en modo IA-32e se activa PAE antes de activar Paginación.
- Si se desactiva PAE en Modo IA-32e
- Si se intenta activar el modo IA-32e desde un código ubicado en un segmento de código de 64 bits.
- Si el registro TR tiene cargada una TSS de 16 bits.

¿Nada mas?

Las Tablas de Paginación deben residir en los primeros 4 Gbytes de memoria antes de habilitar el modo IA-32e.

Condiciones

El procesador chequea los bits ***CR0.PG***, ***CR4.PAE***, y ***IA32_EFER.LME*** para detectar inconsistencias. En cualquiera de los casos siguientes genera una excepción #GP.

- Si se está en modo protegido y se intenta entrar o salir del modo IA-32e con ***CR0.PG*** = 1
- Si en modo IA-32e se activa PAE antes de activar Paginación.
- Si se desactiva PAE en Modo IA-32e
- Si se intenta activar el modo IA-32e desde un código ubicado en un segmento de código de 64 bits.
- Si el registro TR tiene cargada una TSS de 16 bits.

¿Nada mas?

Las Tablas de Paginación deben residir en los primeros 4 Gbytes de memoria antes de habilitar el modo IA-32e.

Condiciones

El procesador chequea los bits ***CR0.PG***, ***CR4.PAE***, y ***IA32_EFER.LME*** para detectar inconsistencias. En cualquiera de los casos siguientes genera una excepción #GP.

- Si se está en modo protegido y se intenta entrar o salir del modo IA-32e con ***CR0.PG*** = 1
- Si en modo IA-32e se activa PAE antes de activar Paginación.
- Si se desactiva PAE en Modo IA-32e
- Si se intenta activar el modo IA-32e desde un código ubicado en un segmento de código de 64 bits.
- Si el registro TR tiene cargada una TSS de 16 bits.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***.

Arrancando directo a 64 bits

Si nuestra idea es habilitar el modo IA-32e directamente, algunas cosas se pueden hacer en Modo Real.

1. En modo real armar una ***GDT*** con al menos un segmento de 64 bit de código y otro de datos.
2. Inicializar el registro ***GDTR*** con los valores de dirección base y límite de la ***GDT***.
3. Activar PAE. Aun en modo Real podemos hacerlo, usando MOV a ***CR4***, siempre antes de activar Paginación.
4. Setear el bit LME del Model Specific Register ***IA32_EFER***. En Modo Real no tiene efecto pero al estar también PAE activado, ni bien entremos a Modo Protegido se activará directamente el modo IA-32e.
5. En ***CR0***, activar al mismo tiempo la paginación y el bit PE.
6. En este punto estamos en Modo Compatibilidad de 16 bits (ya que en la instrucción anterior estábamos en Modo Real), de modo que solo resta un salto far al segmento de 64 bits definido en la ***GDT***