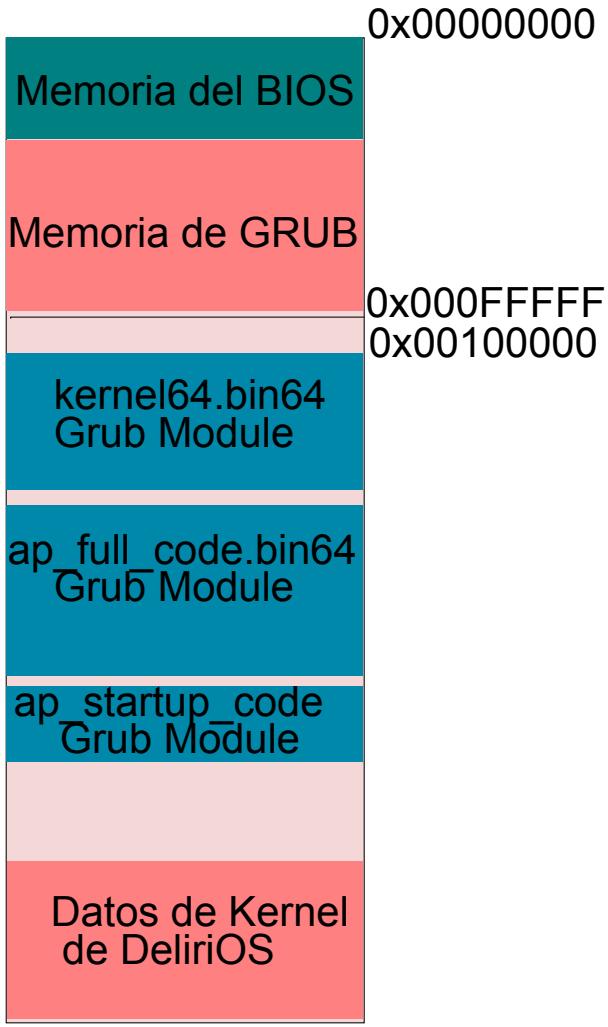
El objetivo de este trabajo práctico fue inicialmente experimentar con la arquitectura intel, realizando un microkernel de 64 bits inicializando multinúcleo. Más tarde en el desarrollo del proyecto se decidió extender el alcance del mismo, realizando algunos experimentos para analizar las posibles mejoras de rendimiento de algoritmos que se pueden paralelizar en varios núcleos, asimismo se realizaron varios enfoques diferentes en la sincronización entre núcleos: espera activa haciendo polling a memoria versus sincronización con interrupciones inter-núcleo que evitan los accesos al bus de memoria. La ganancia que esperamos obtener, es una reducción considerable en el overhead que genera el manejo de varios hilos sobre sistemas operativos utilizando librerías como por ejemplo pthreads.

Utilizamos grub en el nivel mas bajo del booteo para lograr un contexto inicial estable y que posibilitara la ejecución del trabajo practico desde un pendrive usb.
Grub permite iniciar un kernel por medio de una especificación publicada en la web, en la que se detalla un contrato que debe cumplir tanto el kernel a iniciar como grub, entre ellas, un header que debe contener el ejecutable del kernel para ser identificado por grub como un sistema operativo, y por otro lado, las obligaciones que debe cumplir grub al entregarle el control a dicho kernel, es decir, un contexto determinado: selectores de código y datos válidos, registros de propósito general con valores específicos, etc. (`http://www.gnu.org/software/grub/manual/multiboot/multiboot.html`)

Esta revisión de grub no permite cargar ejecutables compilados en 64 bits de manera sencilla, por este motivo decidimos que lo mejor era utilizar a una herramienta provista por grub, llamada carga de módulos, que nos permitió cargar distintas partes no críticas del sistema por encima del primer mega de memoria y tener en un mapa de memoria provisto por grub, las posiciones exactas en la RAM de dichos módulos.

Booteo e integración con grub: Mapa de memoria: Memoria baja y módulos en memoria alta



- **Niveles de booteo del BSP y preparación del entorno de inicio de los AP:**
  1. Grub inicializa la máquina a un estado conocido y otorga el control al loader de nivel 1 del BSP pasándole por parámetros estructuras de grub con información del sistema.
     1. Se realizan validaciones requeridas por la especificación multiboot, verificación de firmas, etc.
     2. Se obtienen de la metadata provista por grub las posiciones de memoria donde están cargados los módulos, ellos son:

        kernel64.bin64:
        : Módulo compilado en formato binario plano de 64 bits que contiene el segundo nivel de booteo del BSP.

        ap_full_code.bin64:
        : Módulo compilado en formato binario plano de 64 bits que contiene el código del segundo nivel de booteo de los Application Processors.

        ap_startup_code:
        : Módulo compilado en formato binario de 32 bits que contiene el código de inicio del AP en modo real y el primer stage de booteo a modo protegido.

Inicializacion por niveles BSP

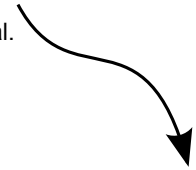
Grub lleva pc a un estado conocido detallado en la especificación multiboot

nivel 1: Puente entre Grub y DeliriOS

* Validaciones de especificacion multiboot.
* Obtención de las posiciones en memoria de los módulos.
* Copia del módulo ap_startup_code a la posición 0x2000.
* Inyección de datos entre módulos.
* Otorgar control al módulo kernel64

nivel 2: Puente entre DeliriOS y modo x64

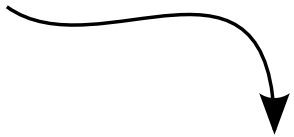
* Asignar GDT, Paginación, Interrupciones, etc.
* Reprogramación del PIC.
* Envío de señal de encendido a los AP's .
* Limpieza de registros de propósito general.
* Seteo de la pila.
* Salto a 64 bits



Inicio completo!

Inicializacion por niveles AP

Ap bootea en 0x2000 en modo real
* Creación de GDT temporal en código
* Salto de modo real a modo protegido
* Salto a loader de nivel 2



nivel 2: Modo protegido a modo x64
* Asignar GDT, IDT, Paginación
* Seteo de la pila.
* Salto a modo x64.

Inicio completo!

Inicialización del Bootstrap Processor: Pasaje desde modo protegido hacia modo legacy x64

Modo Legacy x64: Modelo de segmentación

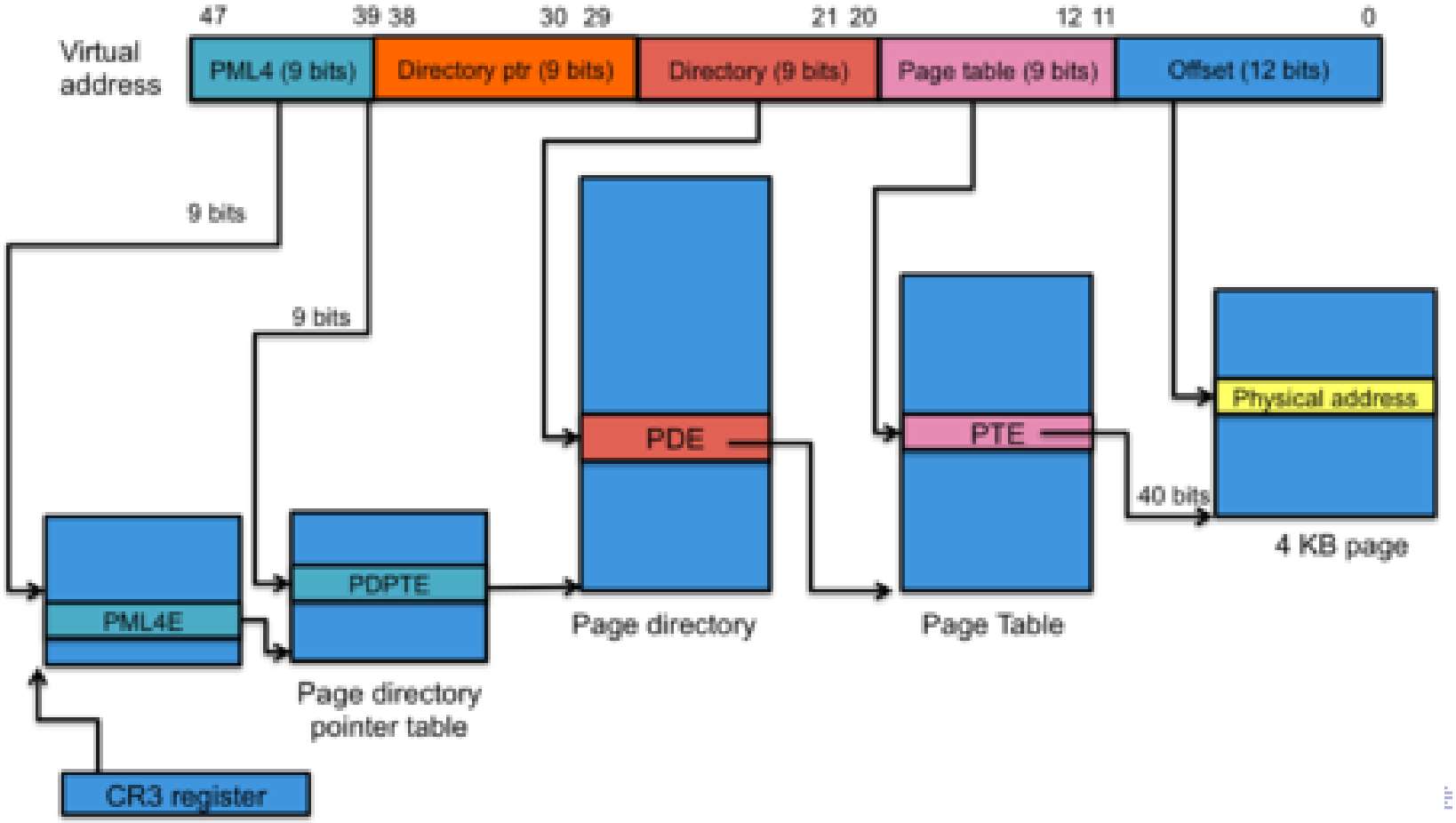
La especificación multiboot nos asegura que estamos en modo protegido, pero no tenemos la certeza de tener asignada una GDT válida, es por esto que asignamos una GDT con 3 descriptores de nivel 0, una para datos y otras dos de código, esta diferenciación de descriptores de código es necesaria para realizar los jump far para pasar de modo real hacia modo protegido y de modo protegido-compatibilidad x64 hacia modo largo x64.

Índice	Descriptor
0	Descriptor nulo
1	Código nivel 0 para 32 bits
2	Código nivel 0 para 64 bits
3	Datos nivel 0 para 32 y 64 bits

Paginacion en 64 bits

x64 requiere habilitar PAE

Como se puede apreciar, tiene mas niveles que la paginacion comun en modo protegido de 32 bits. Se realizo un mapeo identity mapping de los primeros 4gb usando paginas de 4kb.



Verificacion de disponibilidad de x64

Consulta via cpuid

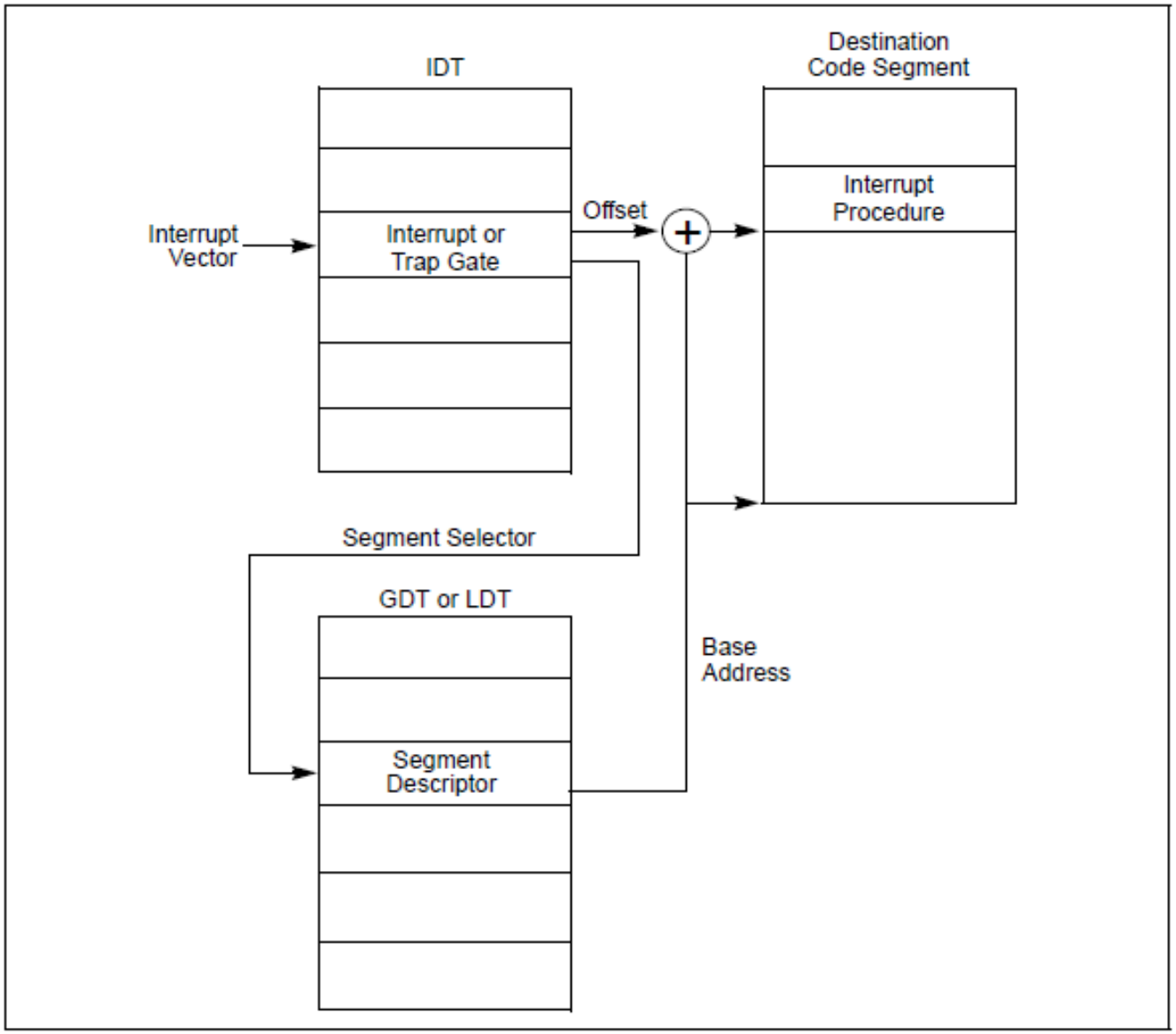
Luego de establecer estas estructuras, realizamos una comprobación vía *cpuid* para saber si está disponible modo x64, de verificarse esta comprobación, encendemos los bits del procesador correspondientes para habilitar dicho modo en el registro CR0.

Pasaje a modo largo x64 nativo

Para pasar de modo compatibilidad a modo nativo de 64 bits, es necesario realizar un salto largo en la ejecución a un descriptor de la GDT de código de 64 bits.
Luego de realizar el salto al segmento de código de x64 de la GDT establecemos un contexto seguro con los registros en cero, seteamos los selectores correspondientes de la GDT y establecemos los punteros a una pila asignada al BSP.

Inicialización del PIC - Captura de excepciones e interrupciones

Enviamos señales al PIC para reprogramarlo de forma tal en la que atienda las interrupciones enmascarables. Luego, asignamos una IDT que captura todas las excepciones e interrupciones y de ser necesario, realiza las acciones correspondientes con su ISR asociada. Particularmente las excepciones son capturadas y mostradas en pantalla y se utiliza la interrupción de reloj para sincronización y esperas, las demás interrupciones son ignoradas.



Interrupt Procedure Call

Multicore: encendido de los AP's

Una vez inicializado el bsp, se procede a inicializar el resto de los procesadores del sistema.
Para lograr esto, primero es necesario que encontrar una estructura de datos llamada MP Floating Pointer Structure, que contiene la información sobre: los demás procesadores; apic (advanced programable interrupt controller); y el I/O apic (análogo al apic, pero se encarga además de rutear interrupciones de input/output a los lapic's).

Búsqueda de la estructura MP Floating Pointer

Esta estructura puede estar en diferentes lugares de memoria, inicialmente se debe realizar la búsqueda dentro del primer kilobyte de la ebda (extended bios data area), de no encontrarse allí, se procede a buscar entre los 639K-640K de memoria. Para encontrar la tabla se busca dentro de esas áreas de memoria la firma de la MP Floating Pointer Structure, la cual es "_MP_".

Comprobación de checksum de la estructura

Una vez encontrada una estructura con esa firma, es necesario realizar una comprobación de checksum para verificar que la misma sea válida. De no ser válida la estructura encontrada, se continúa buscando dentro de las áreas de memoria previamente mencionadas, al agotarse las áreas mencionadas sin éxito en la búsqueda, se concluye que el sistema no soporta multicore.

Habilitación de ICMR y Local APIC

El sistema luego es configurado usando la MP Configuration Table, extraída de la MP Floating Pointer Structure, que contiene en diferentes entradas la información acerca de los diferentes dispositivos utilizados en multicore. Una vez parseadas estas entradas, si la maquina tiene ICMR, se habilita ese registro, y luego se procede a inicializar el apic local del bsp, seteando el vector de interrupciones espurias (interrupciones por ruido de hardware, etc) y habilitando el bit de enable dentro de los registros del local apic. (Registros del local apic especificados en Table 10-1 Local APIC Register Address Map, manual 3 de intel capitulo 10).

Inicialización de los AP's

Una vez finalizada la inicialización del local apic, se debe pedir a la BIOS que setee el warm reset vector a la dirección donde esta localizado el inicio de modo real de los aps($0x2000$). Luego de este paso, se procede a encender los aps usando la información obtenida por la MP Configuration Table.
El proceso de encendido los aps consiste en preparar 3 estructuras de interrupt command register (registro del apic usado para ipis) para luego enviar las ipis especificas de inicio. El primer icr se setea con el delivery mode de INIT, que sea una interrupción por nivel y con la dirección del ap a iniciar. El segundo icr se setea con el delivery mode de INIT_DASSERT, que sea una interrupción por flanco, y que sea enviado a todos los procesadores. El tercer icr se setea con el delivery mode STARTUP, la dirección física de la página de inicio de ejecución del ap shifteado 12 a derecha, y la dirección del ap.

Inicialización de los AP's(cont)

Luego de tener listos los icr, se procede a mandar las ipis de INIT e INIT_DASSERT, luego se espera unos 10 milisegundos, verificando previamente que las ipis se hayan enviado correctamente. Luego de la espera, se procede a enviar la ipi de STARTUP, se espera 10 ms, se verifica que se haya enviado correctamente, se la vuelve a enviar, se realiza otra espera de 10 ms, y se vuelve a verificar. Una vez terminado este proceso, se puede asumir que el ap fue encendido, y se continúa el encendido el resto de los aps encontrados en la MP Configuration Table.

Notas

Nota 1: El proceso de inicialización de los aps se encuentra especificado en detalle en el manual de intel volumen 3, capitulo 8, subsección 4, MULTIPLE-PROCESSOR (MP) INITIALIZATION.
Nota 3: Los identificadores de núcleo no necesariamente son valores numéricos consecutivos.

Multicore: inicialización de modo real a modo nativo x64 de los AP's

Como vimos en la sección anterior, los Application Processors comienzan su ejecución en una posición otra por debajo del primer mega en modo real, nosotros necesitamos hacer saltar la ejecución a una posición conocida por encima del mega que no se solape con estructuras del kernel ni otros módulos, la solución que propusimos es un booteo por etapas.

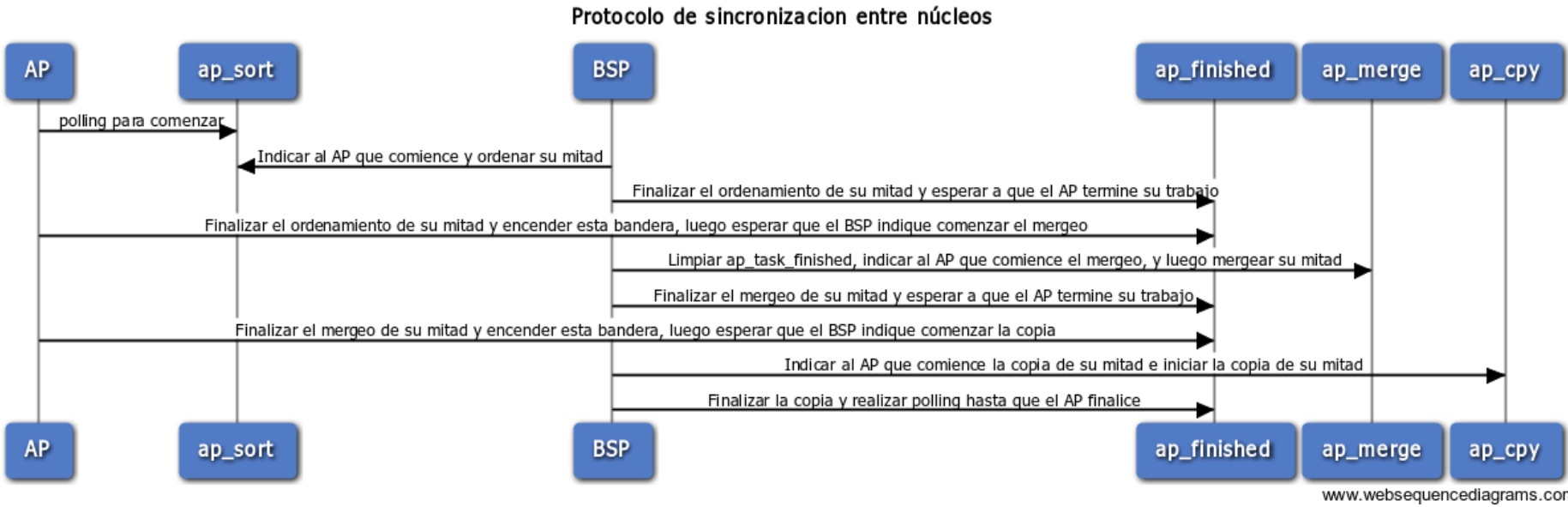
En este primer nivel el núcleo se encuentra en modo real, se inicializa una GDT básica en el mismo código y se salta a modo protegido, esto es necesario para poder direccionar posiciones de memoria por encima del primer mega.

Recordando secciones anteriores, cuando el BSP iniciaba el primer nivel de booteo preparaba el contexto de los APS para inicializar en niveles, en este proceso se inyecta en el código del primer nivel de booteo del AP la posicion de memoria donde esta el segundo nivel de booteo por encima del mega.

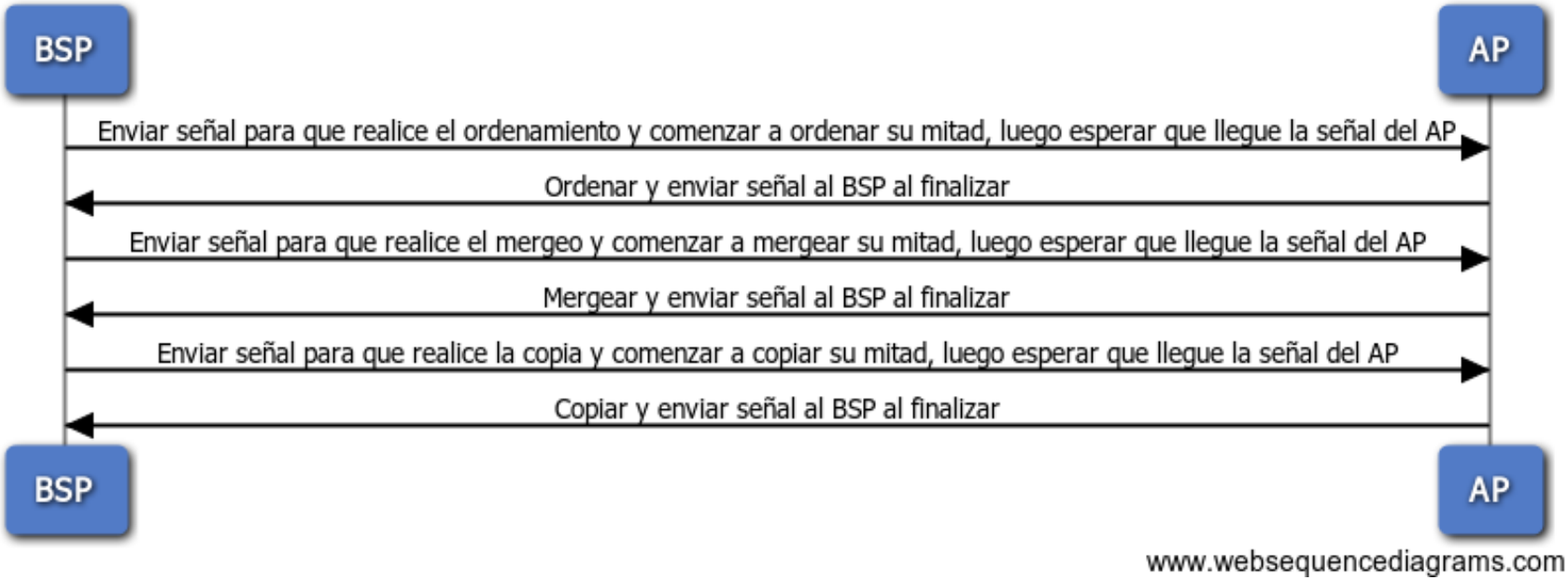
Sorting Paralelo

Se optó en este caso por un algoritmo propio que combinara varios ordenamientos conocidos con el objetivo de hacer mas fácil la paralelización del experimento. En particular lo que se realiza es partir el arreglo en 2 mitades, donde cada core realiza heapsort sobre su mitad asignada y luego una intercalación de los resultados con un algoritmo también paralelo en el cual un core realiza el merge de los máximos y el otro el merge de los mínimos, luego resta copiar los resultados y concluimos. El algoritmo tiene 3 subprocesos

- Sort: Ordenamiento de mitades por cada core
- Merge: Merge paralelizado de las mitades ordenadas
- Copy: Copia del resultado a un buffer de destino


Protocolo de sincronizacion entre núcleos
AP
ap_sort
BSP
ap_finished
ap_merge
ap_cpy
polling para comenzar
Indicar al AP que comience y ordenar su mitad
Finalizar el ordenamiento de su mitad y esperar a que el AP termine su trabajo
Finalizar el ordenamiento de su mitad y encender esta bandera, luego esperar que el BSP indique comenzar el mergeo
Limpiar ap_task_finished, indicar al AP que comience el mergeo, y luego mergear su mitad
Finalizar el mergeo de su mitad y esperar a que el AP termine su trabajo
Finalizar el mergeo de su mitad y encender esta bandera, luego esperar que el BSP indique comenzar la copia
Indicar al AP que comience la copia de su mitad e iniciar la copia de su mitad
Finalizar la copia y realizar polling hasta que el AP finalice
AP
ap_sort
BSP
ap_finished
ap_merge
ap_cpy
www.websequencediagrams.com

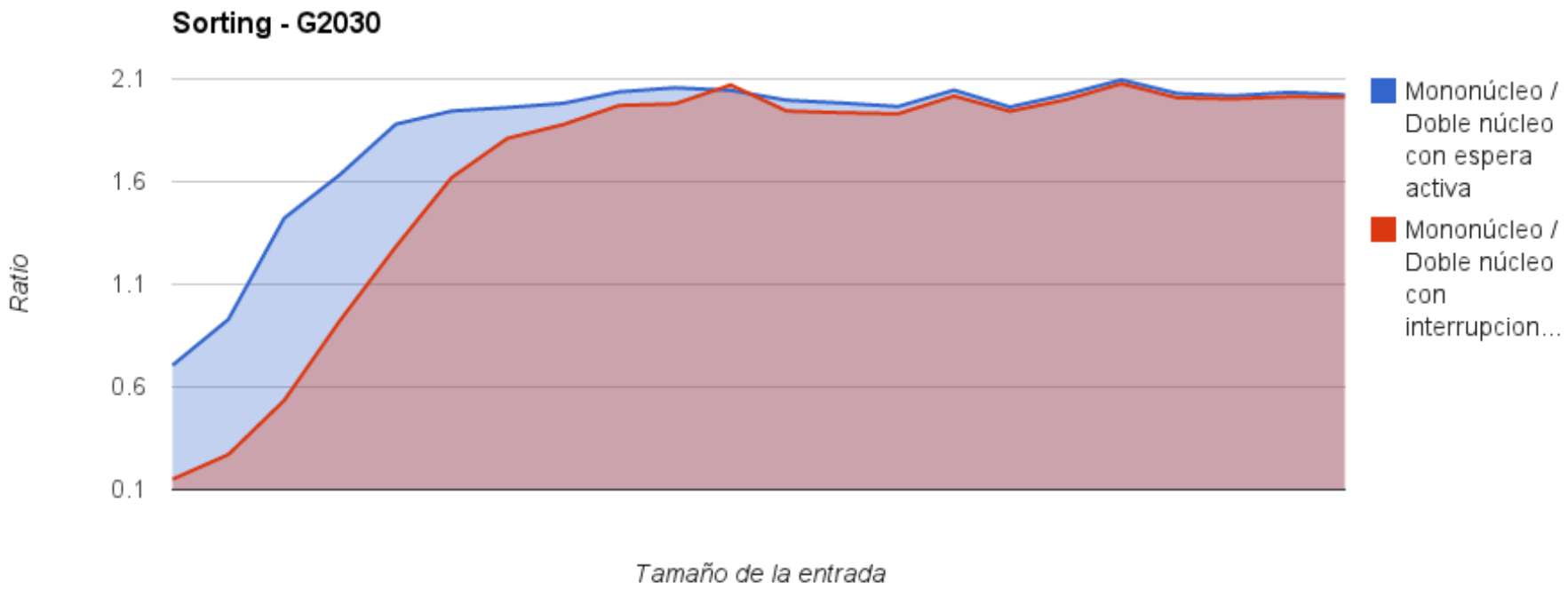
Protocolo de sincronizacion entre núcleos con ipis



Resultados

Intel® Pentium® Processor G2030 - Sorting

Elementos	Ratio Mem/Ipi	Single/DMem	Single/DIpi
2	0.213	0.705	0.150
4	0.291	0.929	0.270
8	0.375	1.422	0.534
16	0.564	1.634	0.923
32	0.682	1.879	1.283
64	0.833	1.944	1.619
128	0.923	1.96	1.810
512	0.967	2.037	1.971
131072	0.987	2.024	1.999
262144	0.991	2.095	2.077
524288	0.989	2.029	2.008
1048576	0.992	2.018	2.003
2097152	0.989	2.034	2.013
4194304	0.994	2.023	2.012


Sorting - G2030
2.1
1.6
1.1
0.6
0.1
Ratio
Tamaño de la entrada
Mononúcleo / Doble núcleo con espera activa
Mononúcleo / Doble núcleo con interrupcion...