Forzando Brutalmente MD5 con computadoras

Carlos Pantelides
@pelucaKiller
http://seguridad-agile.blogspot.com
https://github.com/cpantel

De qué se trata

De cómo atacar por fuerza bruta un hash para obtener su preimagen utilizando distintos hardware y técnicas de programación.

En construcción desde 2017-12, pensada al menos un año antes.

Excusa para aprender y practicar los temas que me interesan.

THE NINJA PROJECT

Objetivos

- Comprender
- Verificación de la comprensión al mostrarlo
- Efectos colaterales útiles
  - Ingeniería inversa
  - Mejor interlocutor
- Out of the Box
- Reinventar la rueda
- Hombros de gigantes

THE NINJA PROJECT

Aplicabilidad

Si siguieras un camino como este, estarías en mejores condiciones para:

- Ataques de fuerza bruta
- Defensa ante ataques de fuerza bruta
- Decisiones de arquitectura
- Optimizaciones generales
- Blockchain

THE NINJA PROJECT

Mis recursos

Libros

- Structured Computer Organization - Andrew S. Tanenbaum – ~2010
- Distributed Systems – Andrew S. Tanenbaum – ~2010
- Computer Architecture: A Quantitative Approach – Hennessy/Patterson – 2017
- Computer Organization and Design: The Hardware/Software Interface ARM Edition – Patterson/Hennessy – 2017

Cursos

- Organización del computador – Saubidet – FIUBA 2007
- Software Performance: Methodology and Techniques – Arzhan Kinzhalin – ECI 2007
- Programación Distribuida y paralela usando MPI – Arzhan Kinzhalin – ECI 2011
- Circuitos Lógicos Programables – Nicolás Alvarez – CESE 2017
- Algunos moocs de Udemy, no recomendables

Décadas de investigación y práctica

Horas y horas de reflexión colectiva

THE NINJA PROJECT

Contexto: Qué es y para qué sirve un hash

Es un función no reversible

“hola que tal?” -> 104 111 108 97 *32* 113 117 101 *32* 116 97 108 63 -> ***1199*** -> 20 -> 2

“Chile 4” -> 67 104 105 108 101 *32* 52 -> ***569*** -> 20 -> 2

Comprobar que los datos no se hayan modificado en tránsito o almacenamiento.

Guardar claves sin conocerlas.

THE NINJA PROJECT

Contexto: Ataques

- Ingeniería social
- Falla criptográfica
- Online
- Offline
  - Diccionario
    - Priorizado
  - fuerza bruta

THE NINJA PROJECT

Descripción MD5

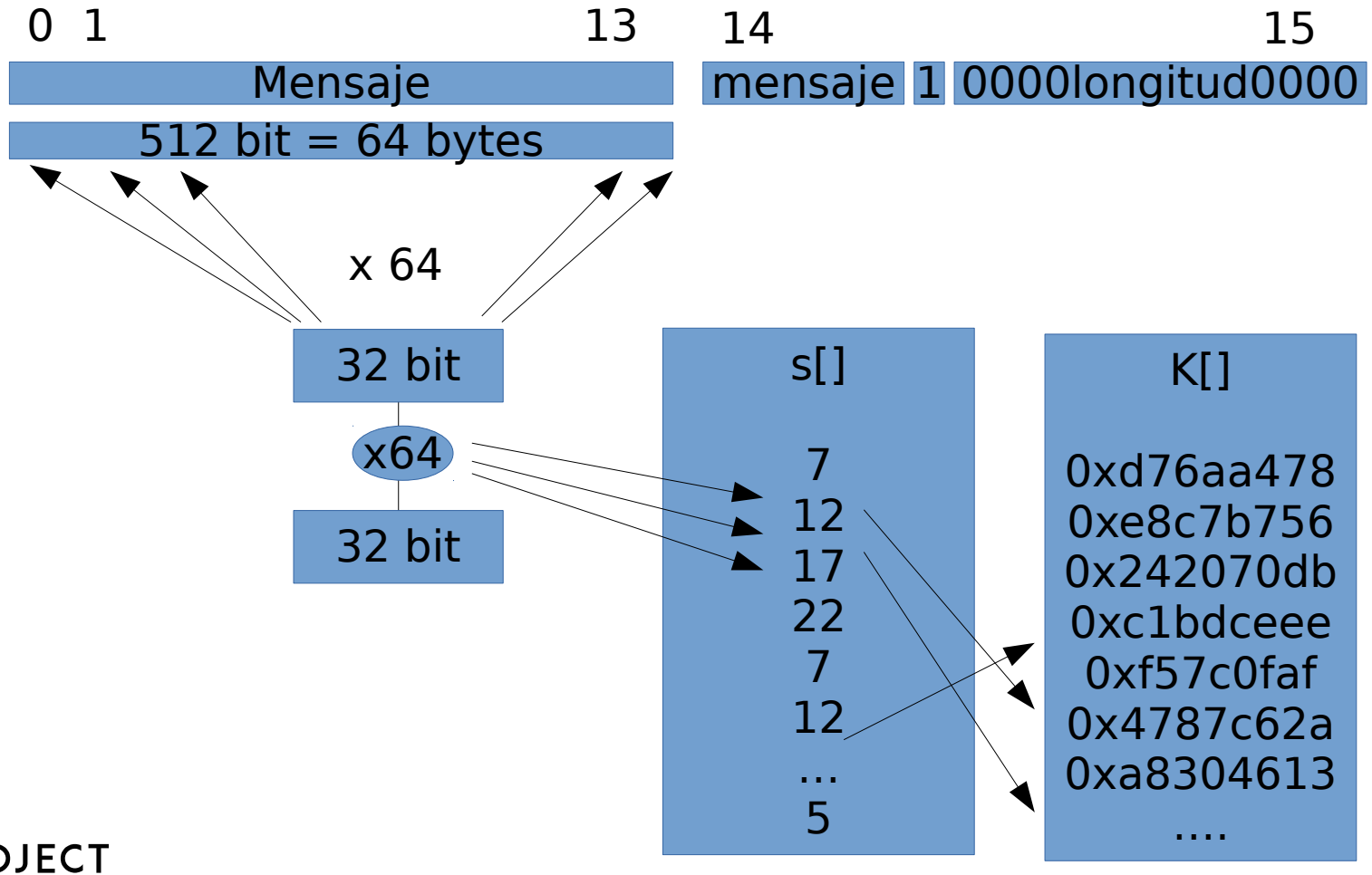
Son 64 rondas en grupos de 16 aplicadas repetidamente a buffers de 512 bits (64 bytes), aplicando un pad para completar.

Se aplica un valor que se calcula con una función que se puede precalcular.

Es obsoleto.

THE NINJA PROJECT

MD5



THE NINJA PROJECT

Optimizaciones propias

Como las claves miden menos de 56 bytes, no hace falta encadenar.

Como las claves miden siempre lo mismo, el pad se calcula una sola vez.

Como es un solo bloque y tiene muchos ceros, se puede hardcodear K.

Se puede hacer loop unroll

THE NINJA PROJECT

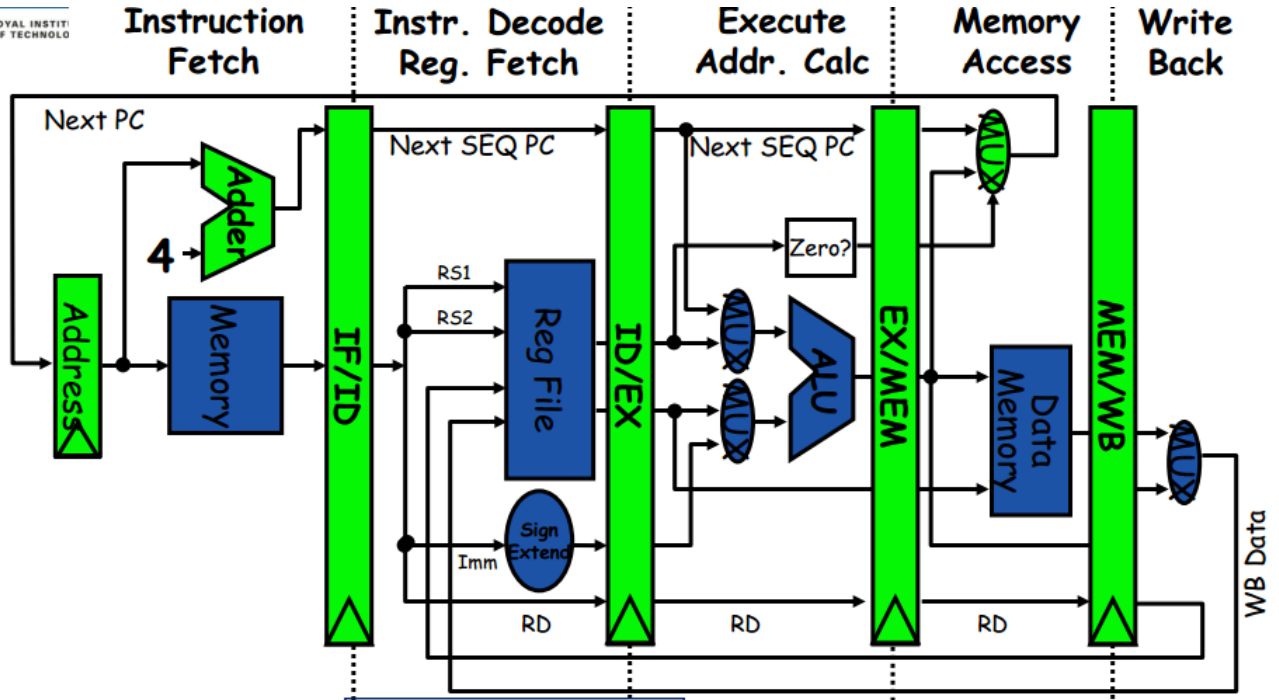
Optimización del flujo → loop unroll

Para cada ronda X de 0 a 63
    Operaciones de la ronda X

Operaciones de la ronda 0
Operaciones de la ronda 1
Operaciones de la ronda…
Operaciones de la ronda 63

THE NINJA PROJECT

Optimizaciones: ILP (Instruction Level Parallelism)




```
IR <= mem[PC];
PC <= PC + 4
```

```
A <= Reg[IR_rs];
B <= Reg[IR_rt]
```

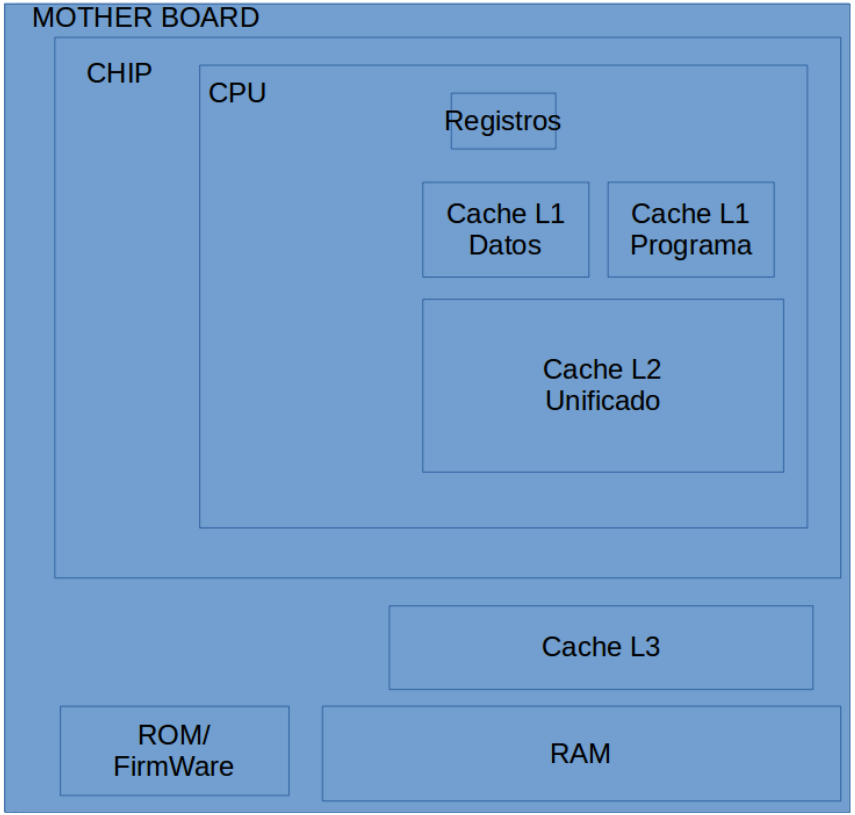
```
rslt <= A op_IRop B
```

```
WB <= rslt
```

```
Reg[IR_rd] <= WB
```

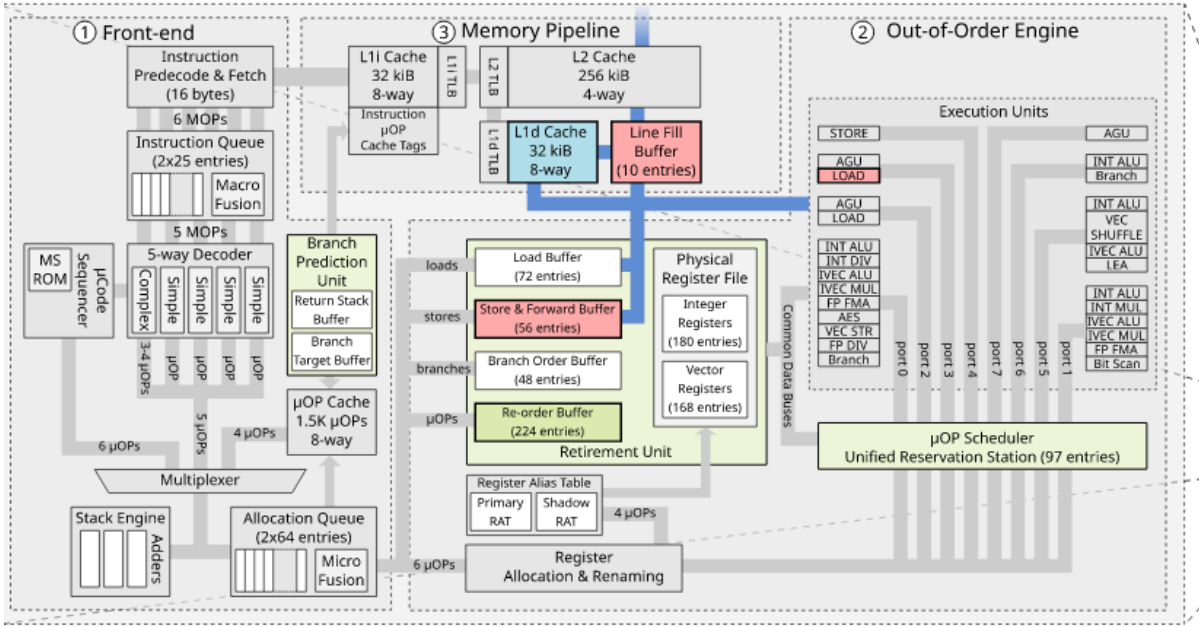
THE NINJA PROJECT

Optimizaciones: registros vs caches vs memoria

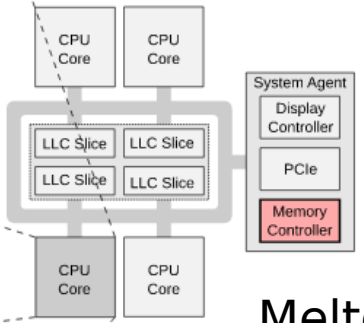


THE NINJA PROJECT

Problemas de seguridad de ILP y caches



https://mdsattacks.com

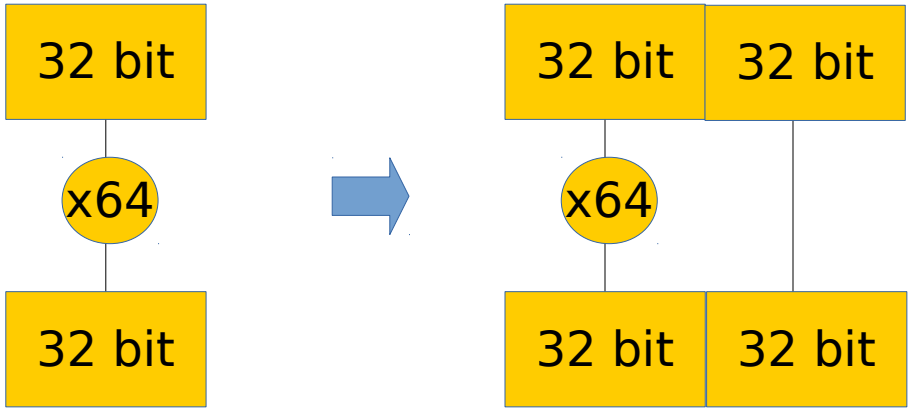


Meltdown
Spectre
Foreshadow
Zombieload
RIDL
Fallout

THE NINJA PROJECT

Hack de 64 bits fallido

Hack de 64 bits fallido



THE NINJ PROJECT

Extensiones: ¿qué?

2018-04-06T19:09:30.523610Z 0 [Note] mysqld (mysqld 5.7.21) starting as process 91 ...
2018-04-06T19:09:30.525944Z 0 [Note] InnoDB: PUNCH HOLE support available
2018-04-06T19:09:30.525957Z 0 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
2018-04-06T19:09:30.525971Z 0 [Note] InnoDB: Uses event mutexes
2018-04-06T19:09:30.525973Z 0 [Note] InnoDB: GCC builtin __atomic_thread_fence() is used for memory barrier
2018-04-06T19:09:30.525976Z 0 [Note] InnoDB: Compressed tables use zlib 1.2.3
2018-04-06T19:09:30.525979Z 0 [Note] InnoDB: Using Linux native AIO
2018-04-06T19:09:30.526182Z 0 [Note] InnoDB: Number of pools: 1
2018-04-06T19:09:30.526289Z 0 [Note] InnoDB: Using CPU crc32 instructions

THE NINJA PROJECT

```
[libx264 @ 0x1d49320] using SAR=1/1
[libx264 @ 0x1d49320] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 AVX2 LZCNT BMI2
[libx264 @ 0x1d49320] profile High, level 3.0
[libx264 @ 0x1d49320] 264 - core 148 r2643 5c65704 - H.264/MPEG-4 AVC codec - Copyleft 2003-2015 - ht
ck=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1
```

THE NINJA PROJECT

SIMD Single Instruction Multiple Data

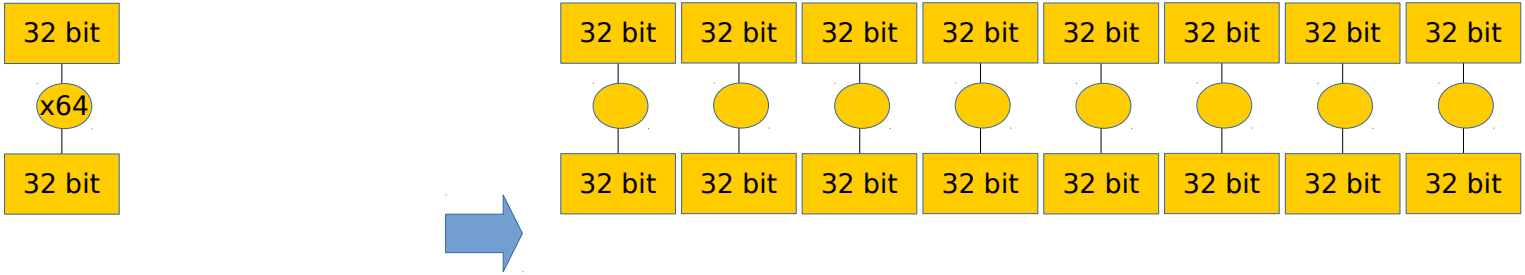
Lo anterior es Single Instruction ***Single*** Data

El hack parece ser SIMD (ojo con el carry)

MMX, SSEx, AVX

THE NINJA PROJECT

Instrucciones vectoriales




```
F = (B & C) | (( ~ B) & D);
```

```
__m256i NotB = _mm256_andnot_si256(B,_mm256_set1_epi32(0xffffffff));

F = _mm256_or_si256(
                _mm256_and_si256(B,C),
                _mm256_and_si256(D,NotB)
);
```

THE NINJA PROJECT

SIMD Single Instruction Multiple Data

Intrinsics

```
#include <immintrin.h>

uint32_t M[LANES][BUFFER_LEN];

...

__m256i a0 = _mm256_set1_epi32(0x67452301);

__m256i A = _mm256_set_epi32(0x0360ac33,0x330dab62,0x4300ab66,0x6300ab33,0xa300a9d2,0xf3003c32,0x2300ab23,0x03433b31);

a0 = _mm256_add_epi32(a0, A);
```

- https://software.intel.com/sites/landingpage/IntrinsicsGuide/

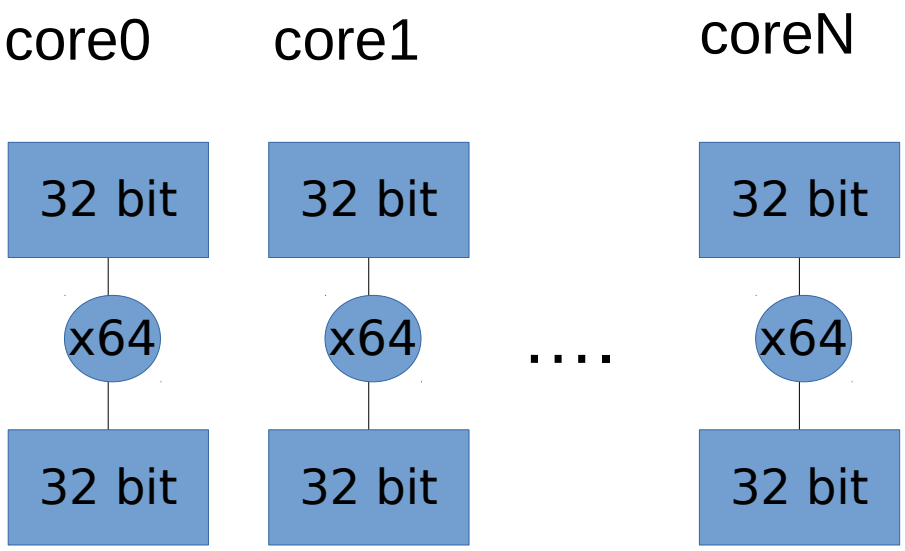
THE NINJA PROJECT

Forks y batch

Se pueden ejecutar múltiples instancias de un mismo programa

Multithread

Se pueden ejecutar múltiples instancias del una misma sección de código compartiendo la memoria.



THE NINJA PROJECT

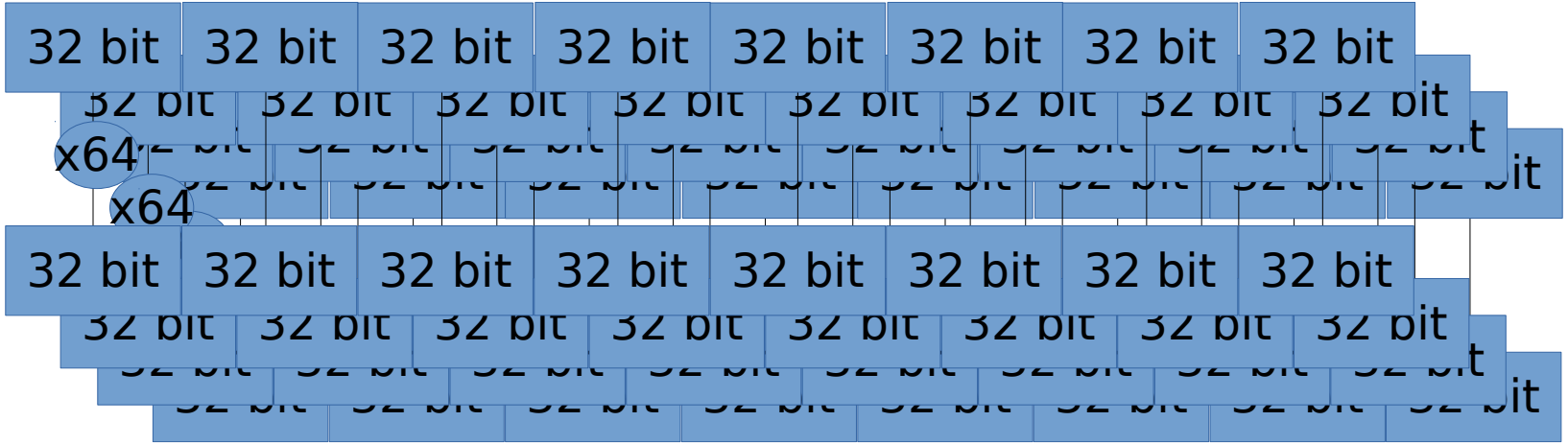
La diferencia entre forks y threads

```
pid_t pid = 0;

Data data[] = { {0,';','J',0},
                {1,'K','Z',0},
                {2,'[','j',0},
                {3,'k','z',0}
              };

int i;
for (i = 0 ; i < 4 ; ++i) {
  if ( (pid == fork() ) != 0) {
     data[i].pid = pid;
     continue;
  } else {
    uint64_t hashcount = data[i].hashcount;
    char init          = data[i].init;
    char stop          = data[i].stop;
     …
    return 0;
  }

}
for (i = 0 ; i < 4 ; ++i) {
  waitpid(data[i].pid);
}
```

```
pthread_t tid[4];

Data data[] = { {0,';','J',argv[1]},
                {1,'K','Z',argv[1]},
                {2,'[','j',argv[1]},
                {3,'k','z',argv[1]}
              };

for (int i = 0 ; i < 4 ; ++i) {
   pthread_create(&(tid[i]), NULL, &doIt, (void *) &data[i]);
}

for (int i = 0 ; i < 4 ; ++i) {
   pthread_join(tid[i], NULL);
}


void * doIt(void * args) {
   uint64_t hashcount = ((Data *) args)->hashcount;
   char init          = ((Data *) args)->init;
   char stop          = ((Data *) args)->stop;
   char * hash          = ((Data *) args)->hash;

   …
   pthread_exit(NULL);
}
```

THE NINJ PROJECT

Multithread/Forks + SIMD



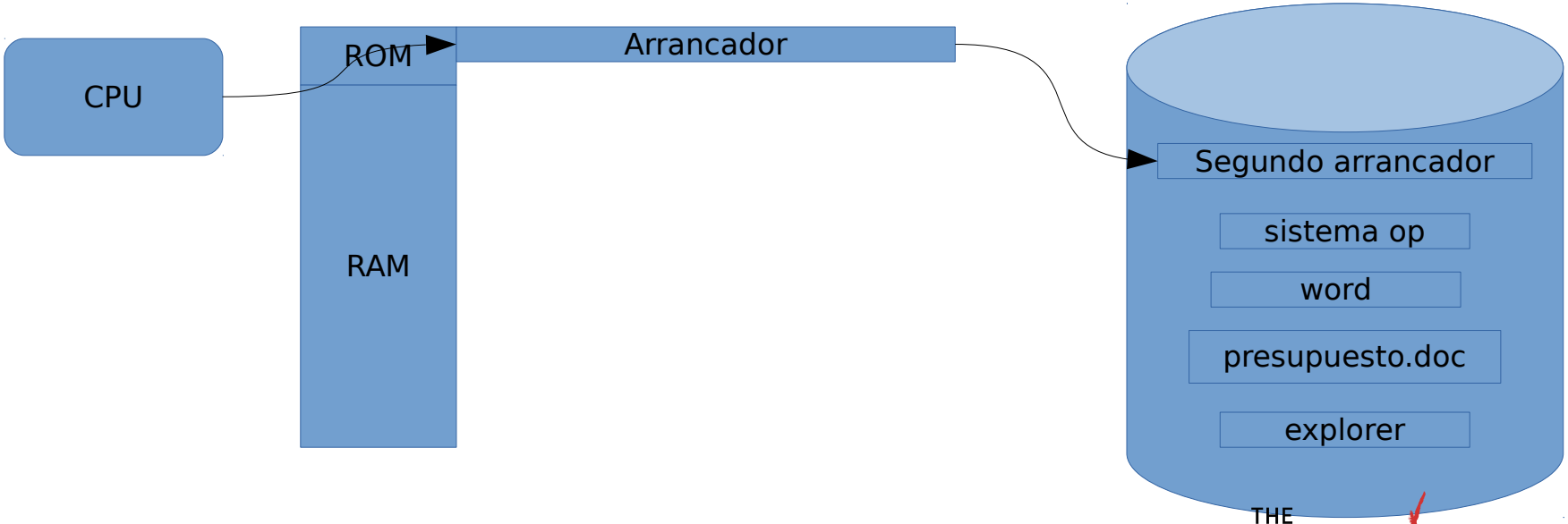
THE NINJ PROJECT

SuperComputadoras: MPI Message Passing Interface

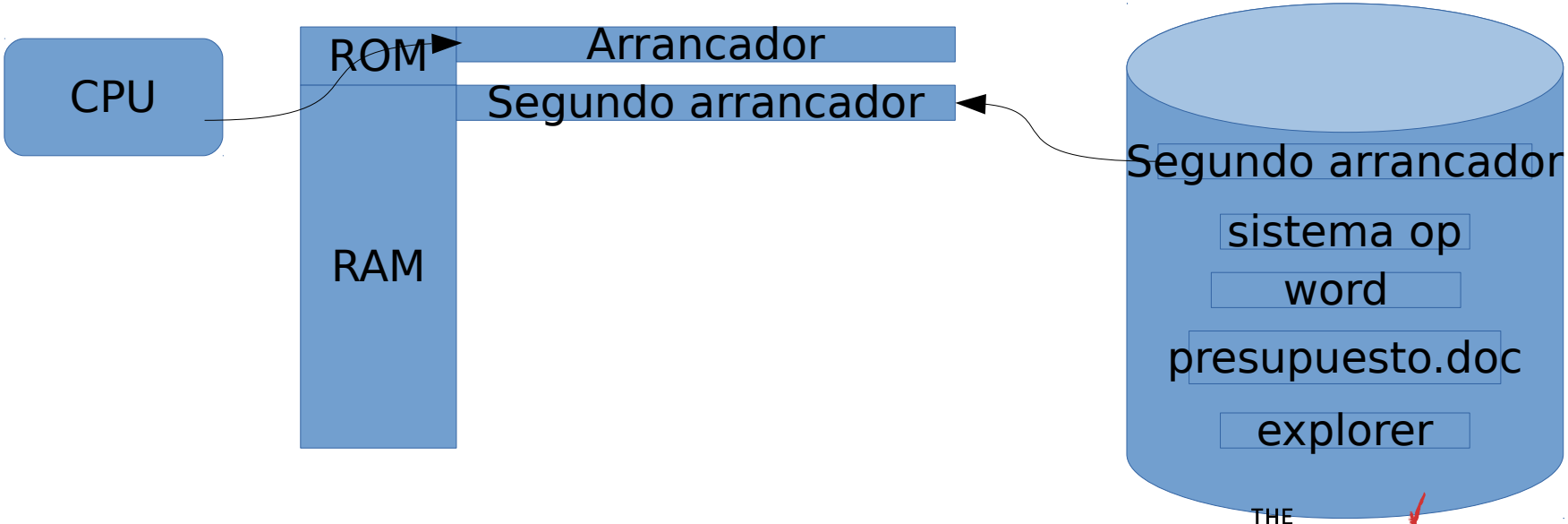
- En que se parece a un multicore:
  - Procesadores “completos”
  - Cada uno podría correr un sistema operativo
- En que se parece a GPU
  - Distinta arquitectura, ISA (distintos compiladores)
- En que se distingue de GPU
  - Como se comunican los cores

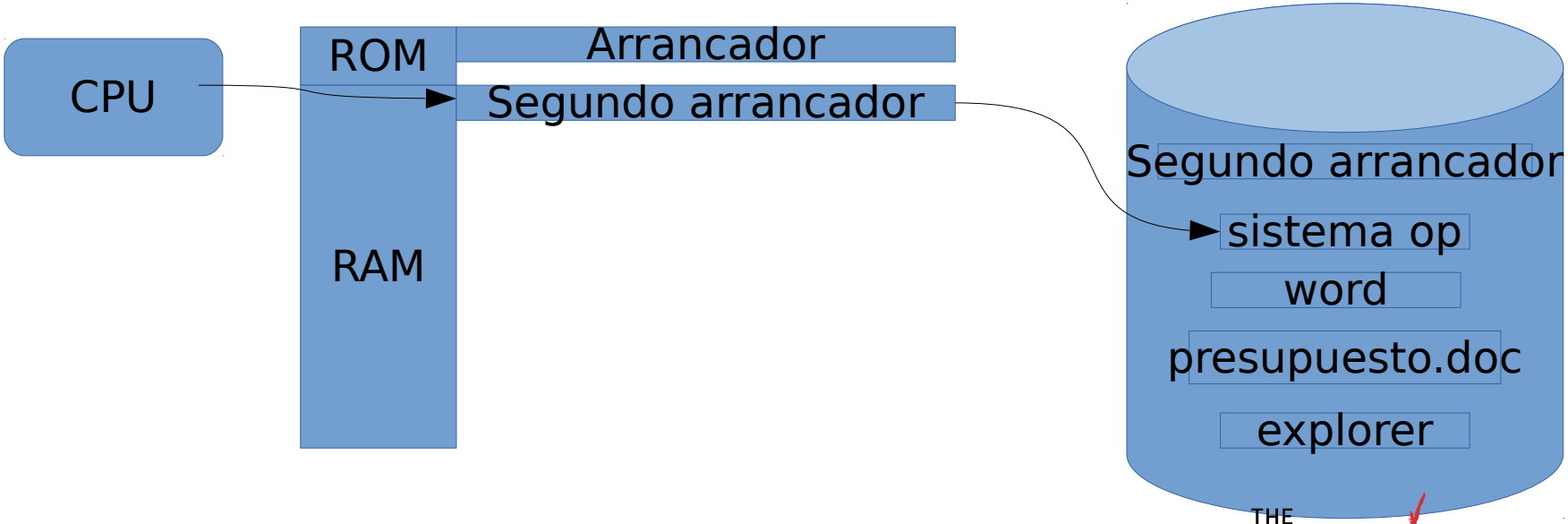
THE NINJA PROJECT

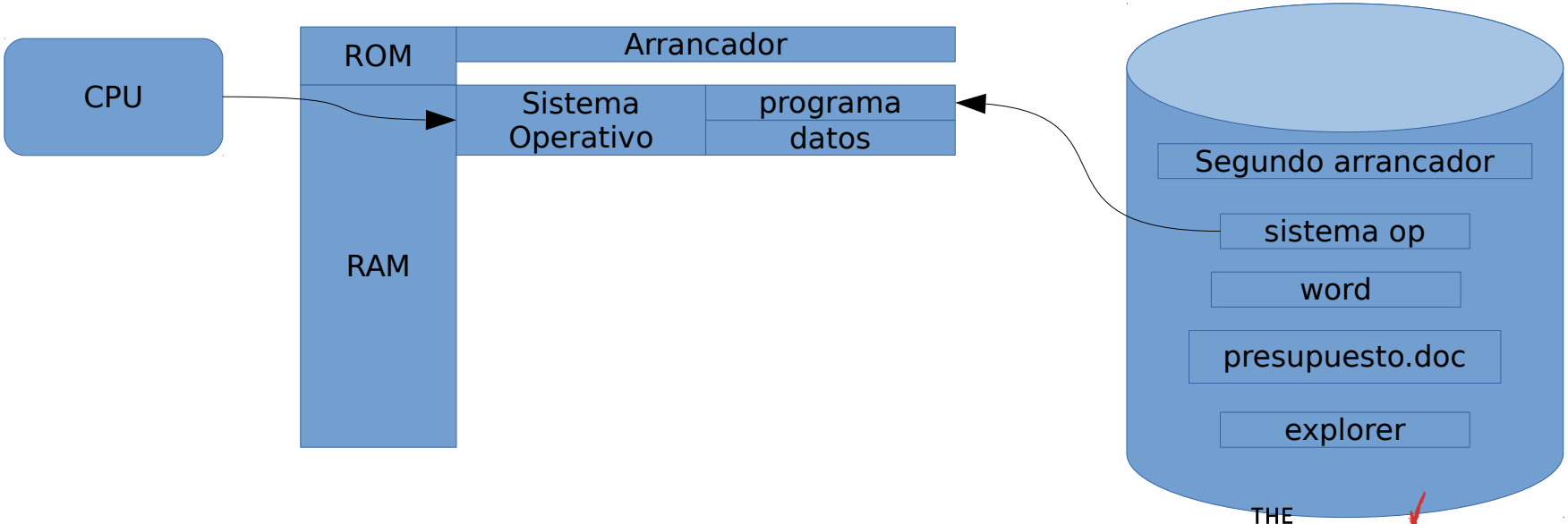
Cómo arranca una compu?

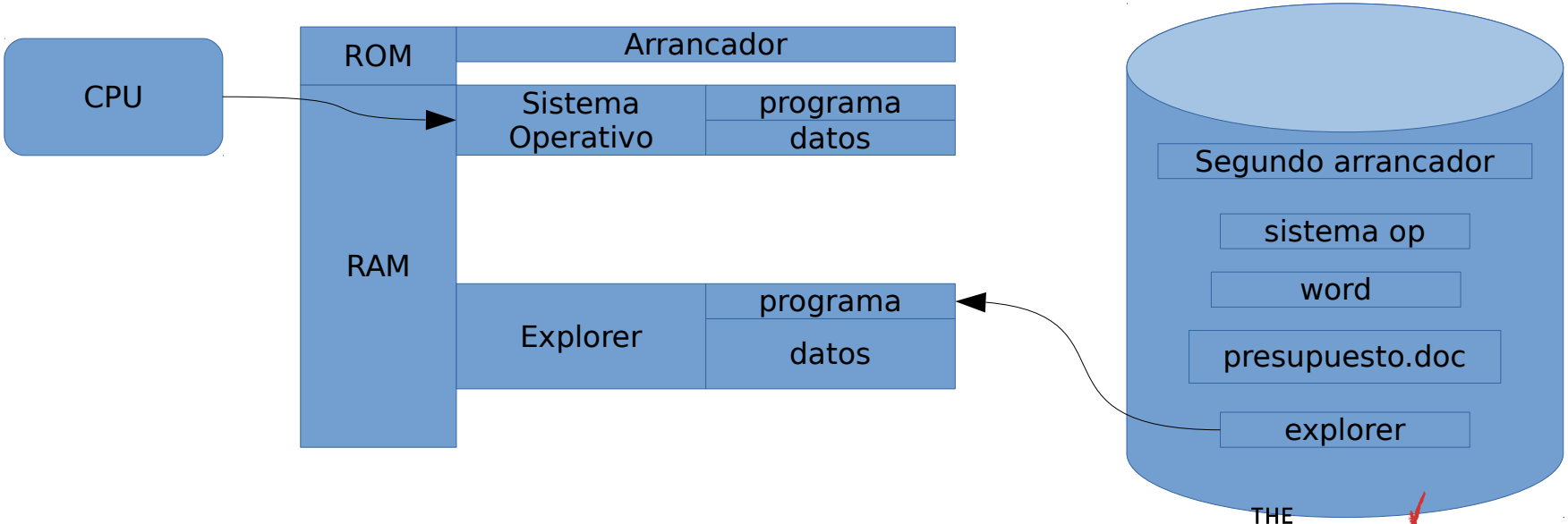


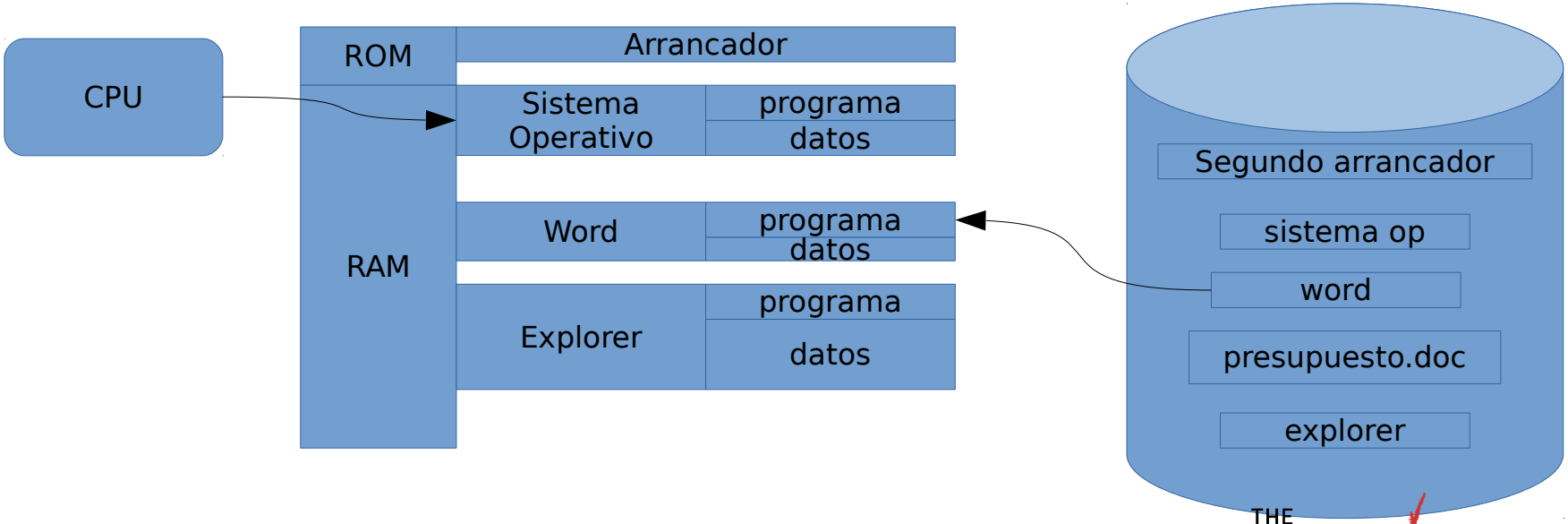
THE NINJA PROJECT

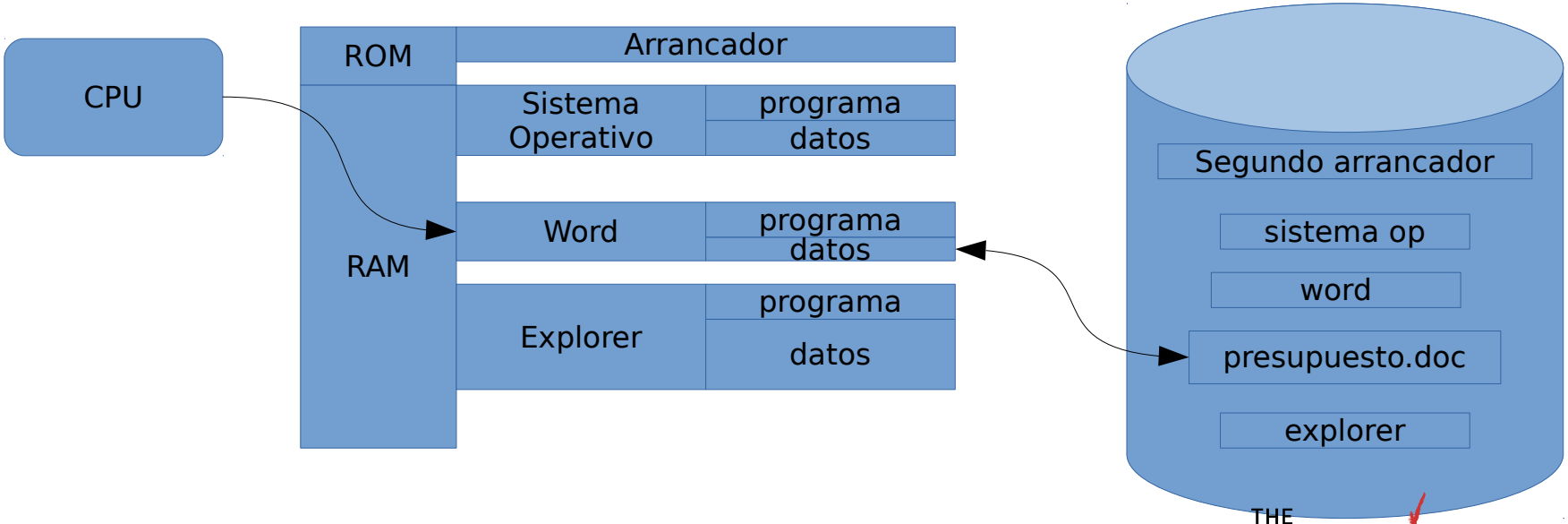


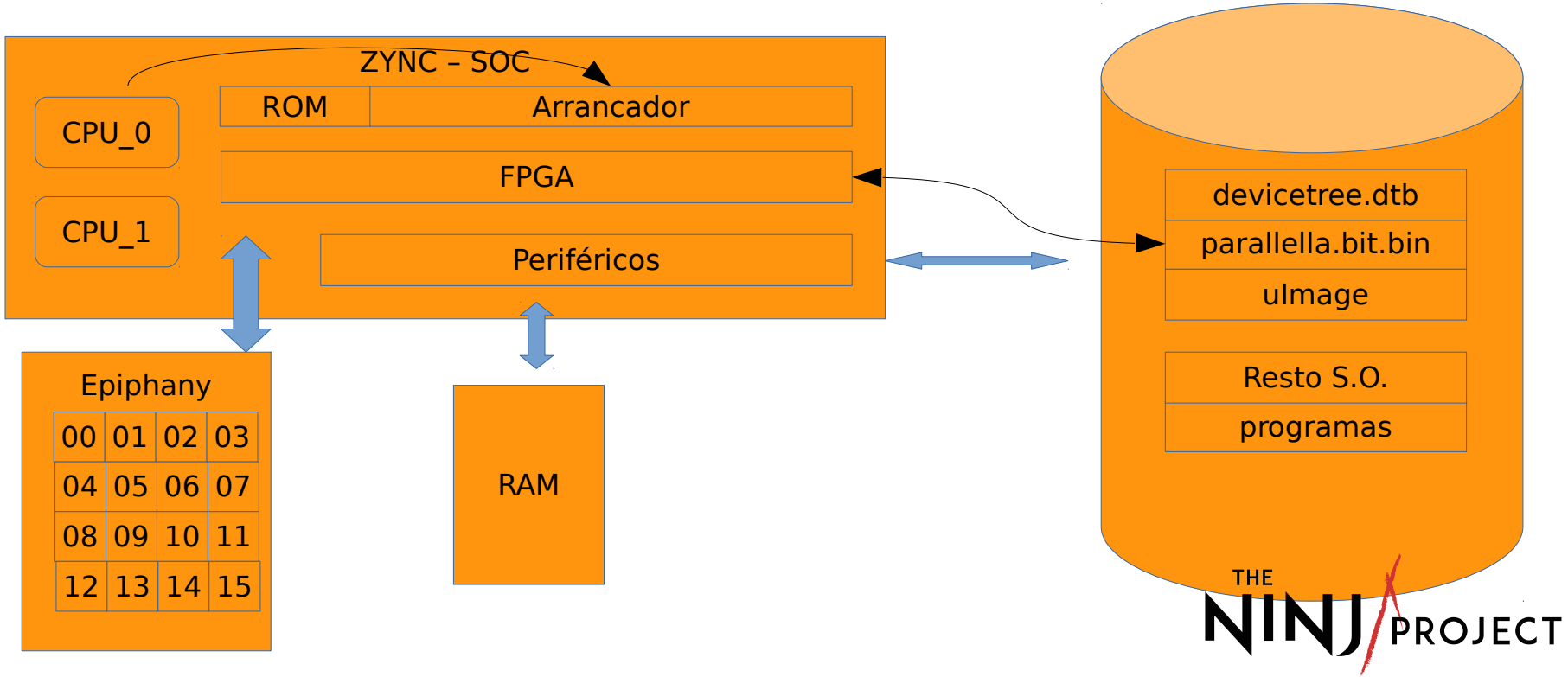


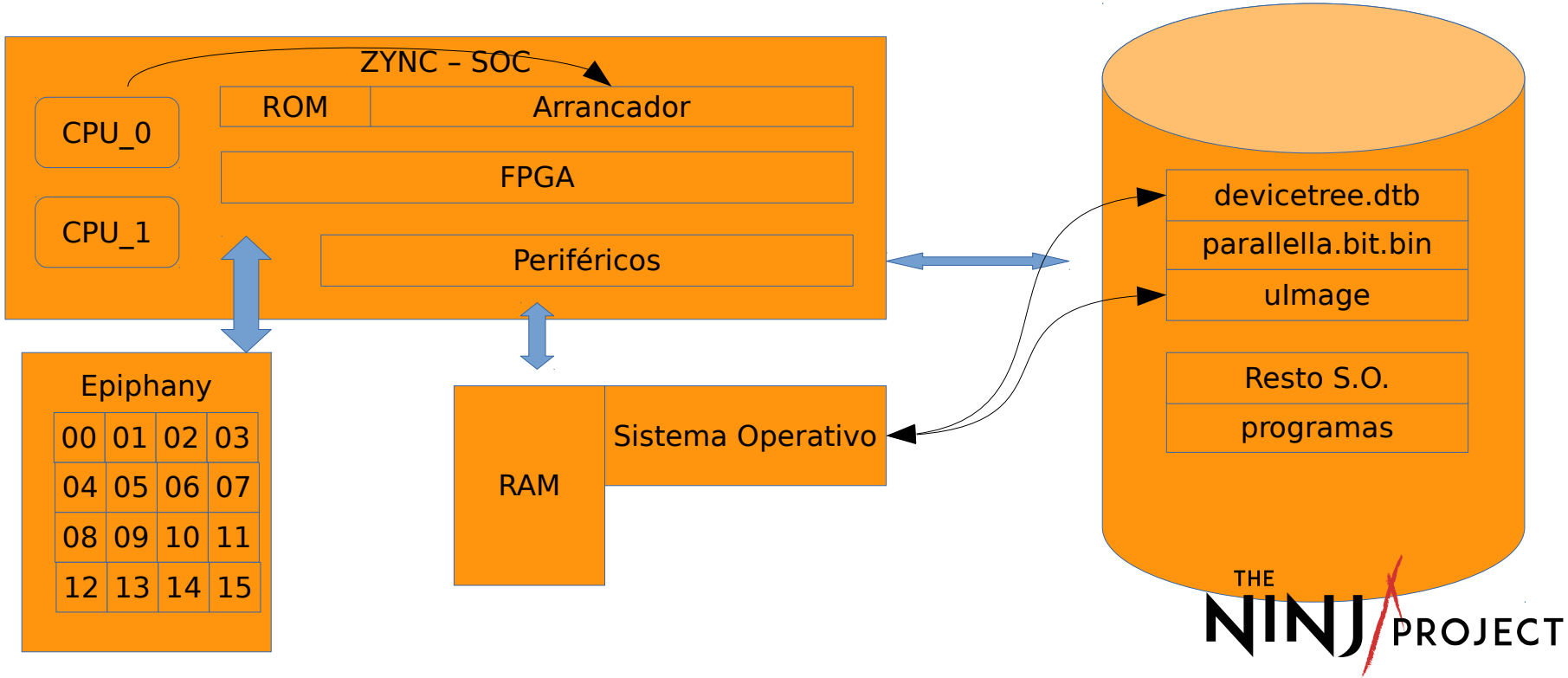


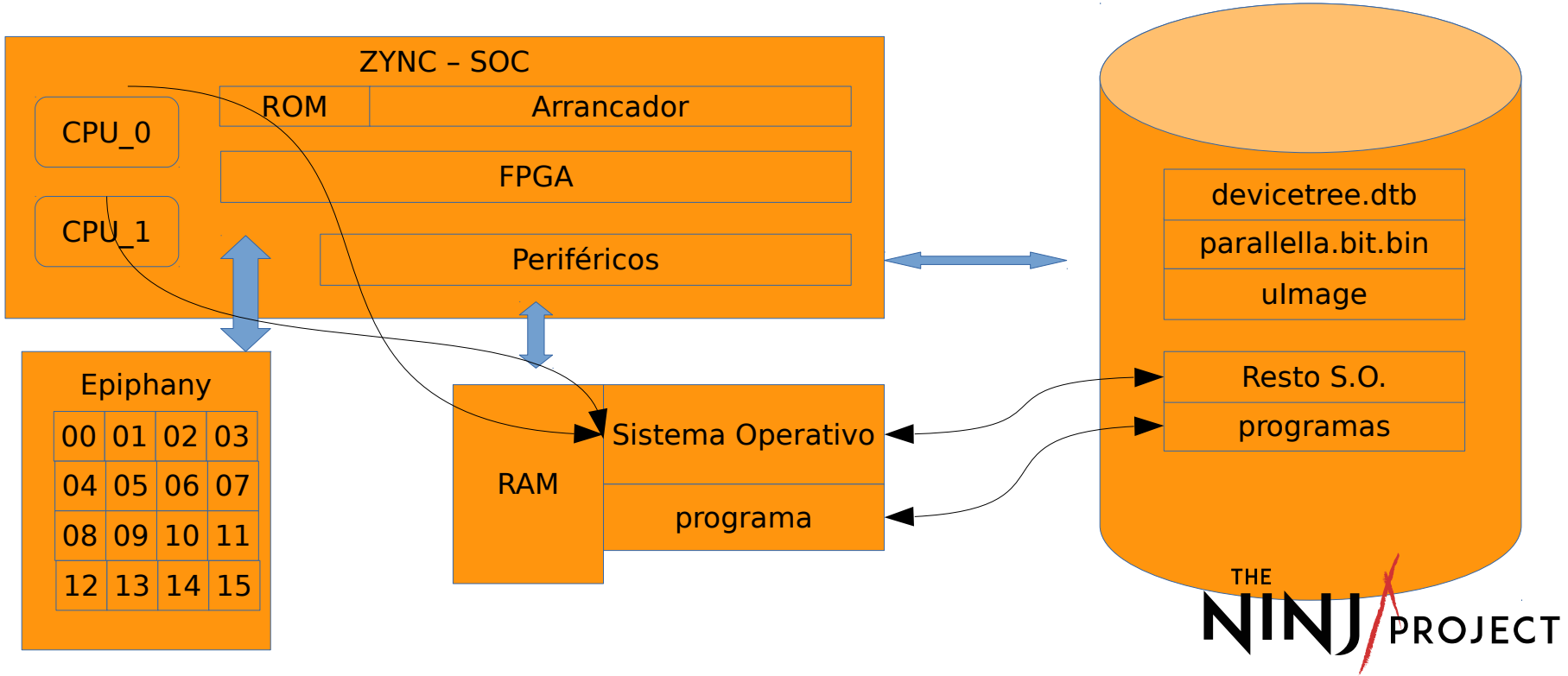


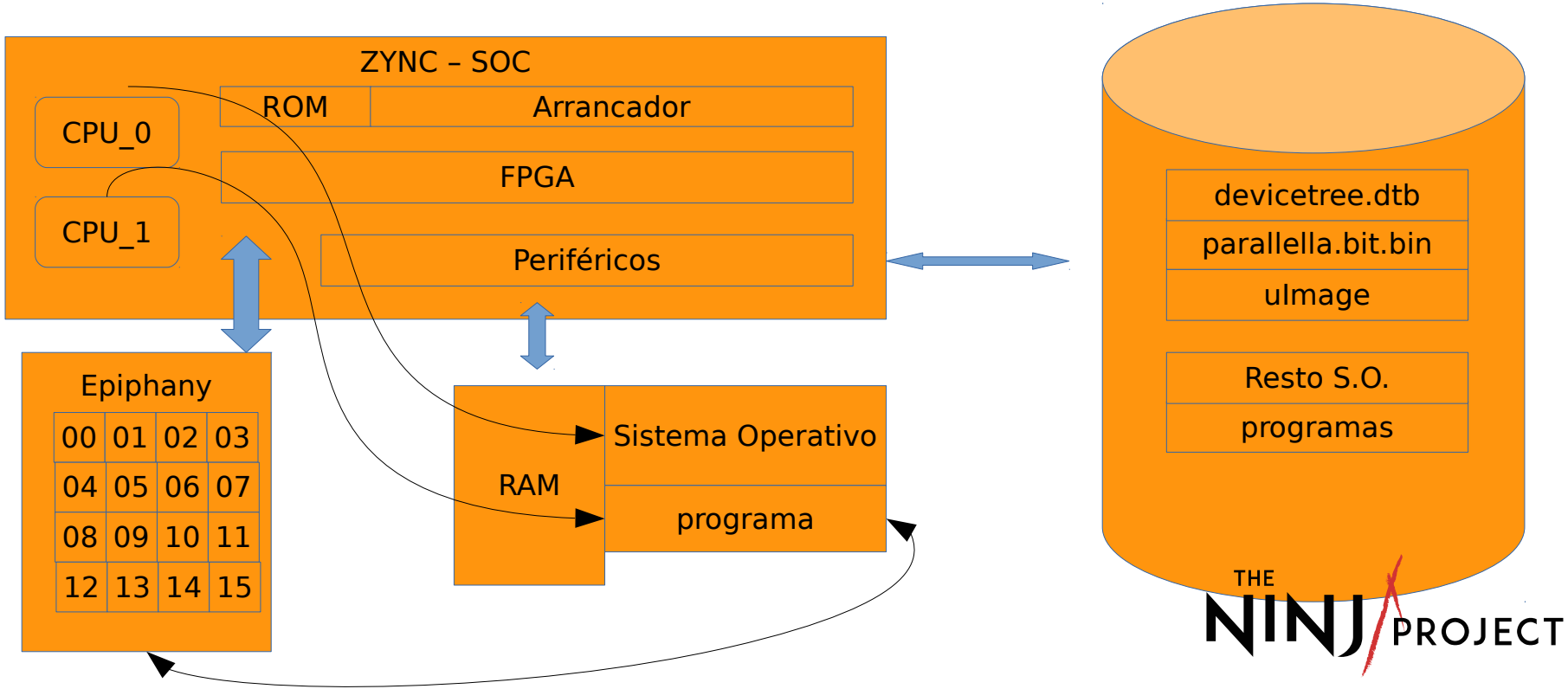


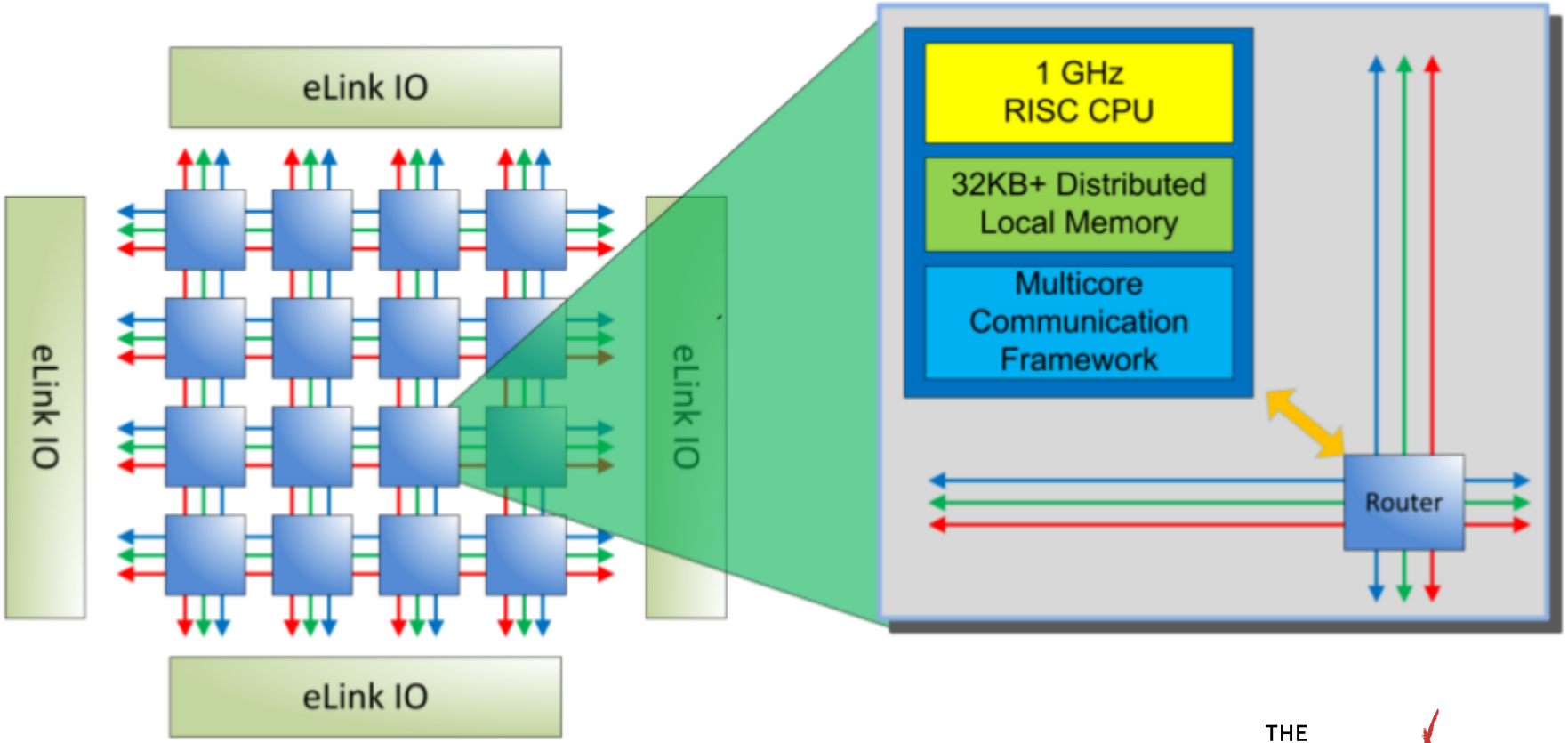



ZYNC – SOC
CPU_0
CPU_1
ROM
Arrancador
FPGA
Periféricos
Epiphany
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15
RAM
devicetree.dtb
parallella.bit.bin
uImage
Resto S.O.
programas
THE NINJA PROJECT

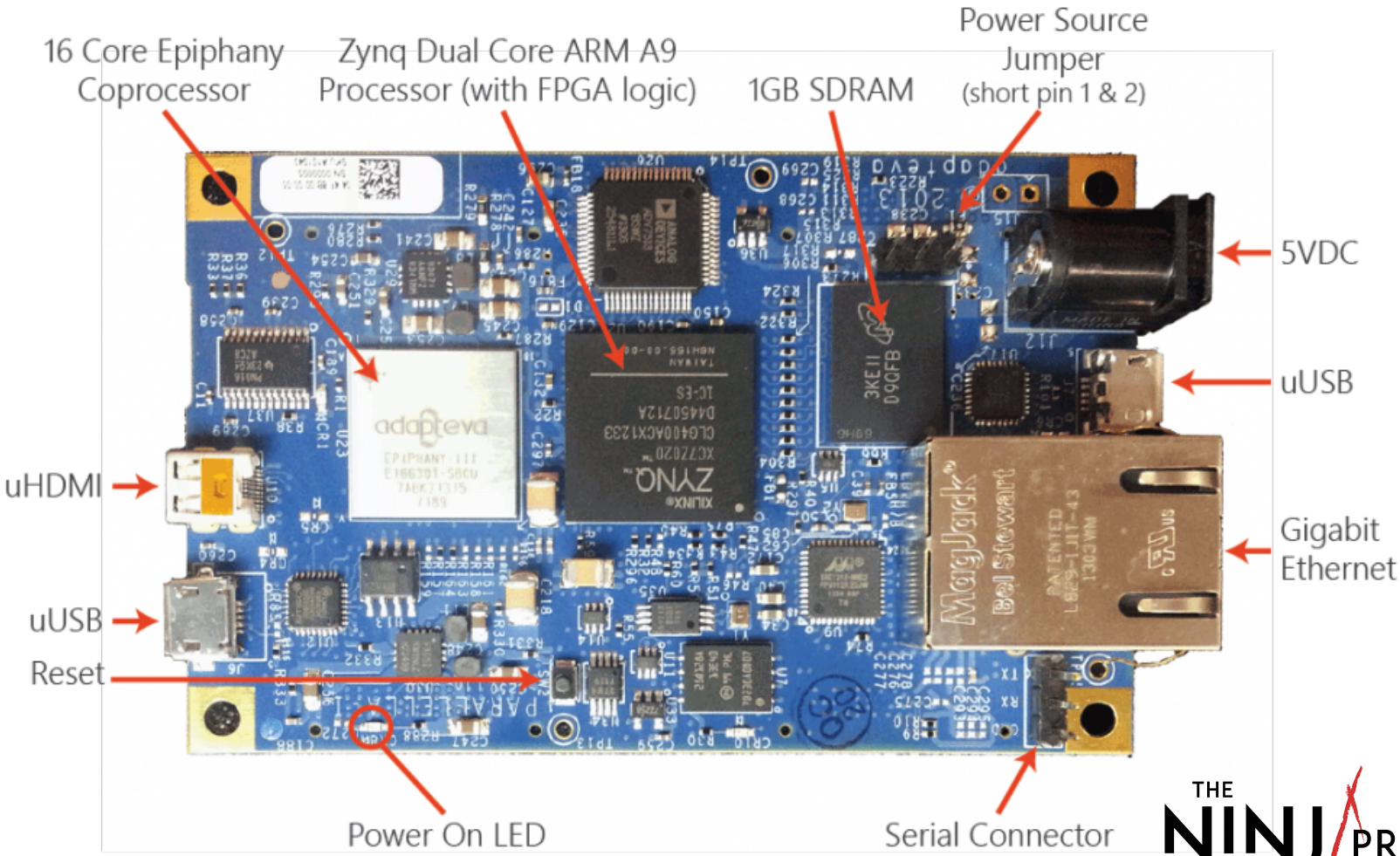

ZYNC – SOC
CPU_0
CPU_1
ROM
Arrancador
FPGA
Periféricos
Epiphany
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15
RAM
Sistema Operativo
devicetree.dtb
parallella.bit.bin
uImage
Resto S.O.
programas
THE NINJA PROJECT


ZYNC – SOC
CPU_0
CPU_1
ROM
Arrancador
FPGA
Periféricos
Epiphany
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15
RAM
Sistema Operativo
programa
devicetree.dtb
parallella.bit.bin
uImage
Resto S.O.
programas
THE NINJA PROJECT

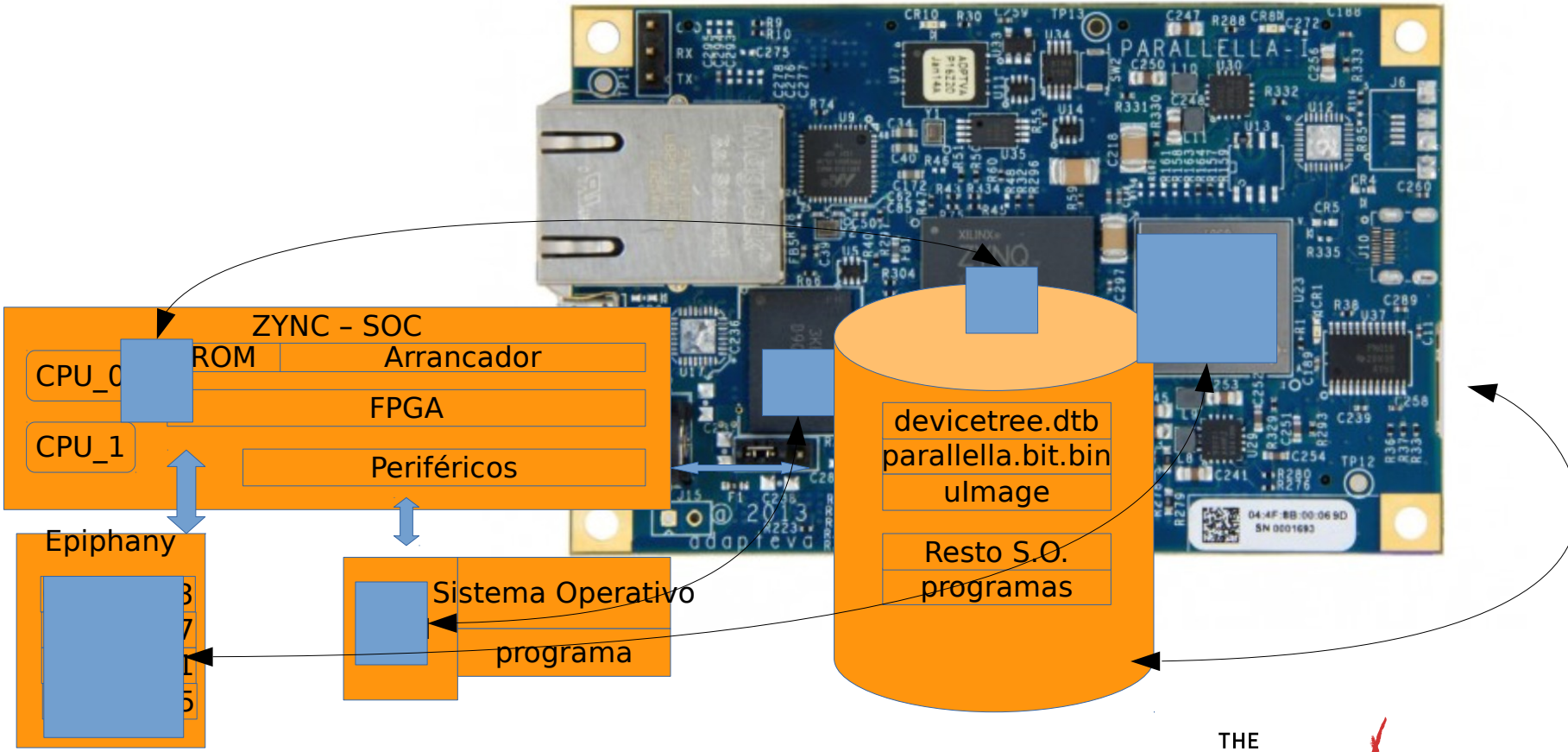

ZYNC – SOC
CPU_0
CPU_1
ROM
Arrancador
FPGA
Periféricos
Epiphany
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15
RAM
Sistema Operativo
programa
devicetree.dtb
parallella.bit.bin
uImage
Resto S.O.
programas
THE NINJA PROJECT



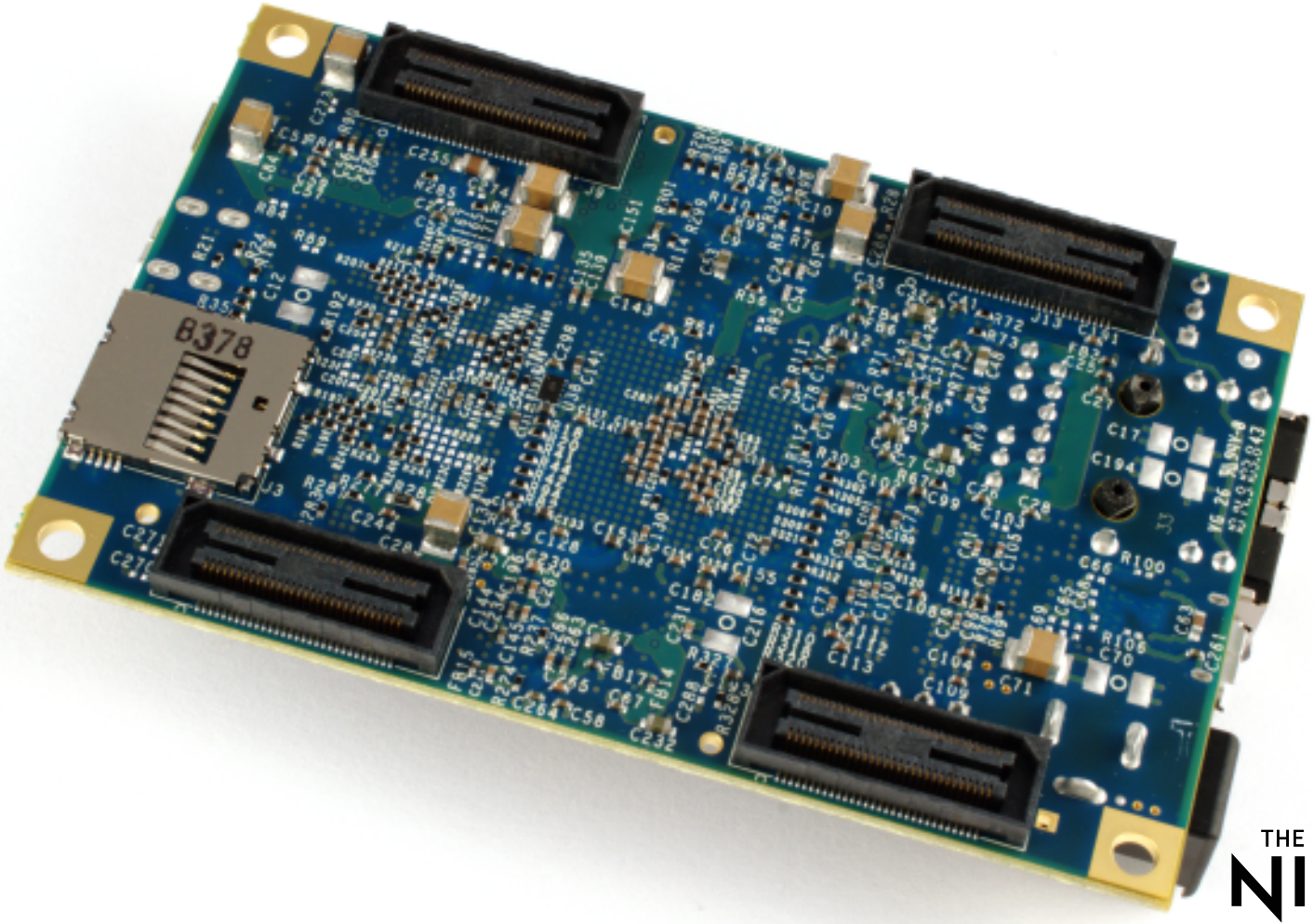
THE NINJ PROJECT



THE NINJ PROJECT



THE NINJ PROJECT



THE NINJ PROJECT

GPGPU General Purpose computing on Graphic Processor Units

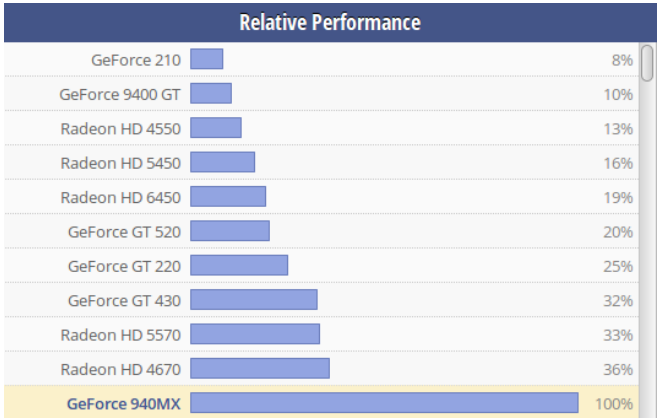
Qué es GPU y de dónde salió (Video juegos…)

- TMS3010
- Silicon Graphics
  - 3dfx
  - Nvidia
- Ati
- Intel
- Broadcom → raspberry PI

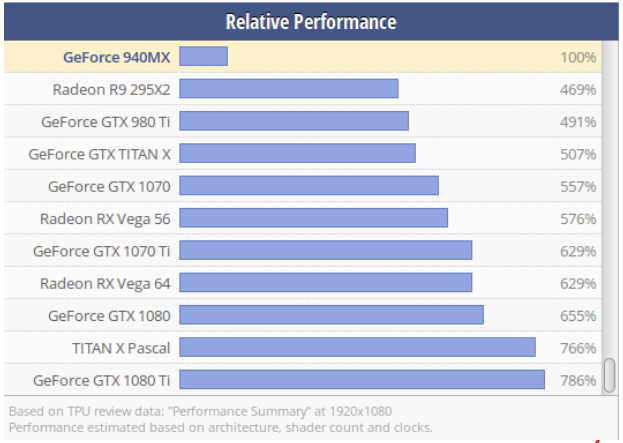
- OpenGL/DirectX → CUDA/OpenCL

THE NINJ PROJECT

GPGPU General Purpose computing on Graphic Processor Units



Utilizaré GeForce 940MX con CUDA, que es específico de NVidia



Dice tener un GM108 graphic processor, con 384 cores a 1Ghz, Maxwell

THE NINJA PROJECT

GPGPU General Purpose computing on Graphic Processor Units

Diferencia con lo anterior:

Ahora hay más gestión que programación.

El procesamiento es más lento, 30 segundos, pero es masivo.

THE NINJA PROJECT

¿Recuerdan forks y pthreads?

```
pid_t pid = 0;

Data data[] = { {0,';','J',0},
                {1,'K','Z',0},
                {2,'[','j',0},
                {3,'k','z',0}
              };

int i;
for (i = 0 ; i < 4 ; ++i) {
  if ( (pid == fork() ) != 0) {
     data[i].pid = pid;
     continue;
  } else {
    uint64_t hashcount = data[i].hashcount;
    char init          = data[i].init;
    char stop          = data[i].stop;
     …
    return 0;
  }

}
for (i = 0 ; i < 4 ; ++i) {
  waitpid(data[i].pid);
}
```

```
pthread_t tid[4];

Data data[] = { {0,';','J',argv[1]},
                {1,'K','Z',argv[1]},
                {2,'[','j',argv[1]},
                {3,'k','z',argv[1]}
              };

for (int i = 0 ; i < 4 ; ++i) {
   pthread_create(&(tid[i]), NULL, &doIt, (void *) &data[i]);
}

for (int i = 0 ; i < 4 ; ++i) {
   pthread_join(tid[i], NULL);
}


void * doIt(void * args) {
   uint64_t hashcount = ((Data *) args)->hashcount;
   char init          = ((Data *) args)->init;
   char stop          = ((Data *) args)->stop;
   char * hash          = ((Data *) args)->hash;

   …
   pthread_exit(NULL);
}
```

THE NINJ PROJECT

Esto es CUDA

```
uint32_t M[BUFFER_LEN];
uint32_t * dev_M;
pad("012345", M);

// copy everything to cuda memory
cudaMalloc( (void**)&dev_M , BUFFER_LEN );
cudaMalloc( (void**)&dev_msg , 9 );
cudaMemcpy( (void*) dev_M, (void*) M, BUFFER_LEN ,cudaMemcpyHostToDevice ) ;

find<<<blocks, threads>>>( hostA, hostB, hostC, hostD, dev_M, dev_msg );

// copy answer from cuda memory
cudaMemcpy( msg, dev_msg, 8,cudaMemcpyDeviceToHost ) ;

if (msg[0] != 0) {
  printf("<<< FOUND KEY : %s >>>\n", msg);
}
```

```
__global__ void find( uint32_t hostA, uint32_t hostB, uint32_t hostC, uint32_t hostD,
                      uint32_t * outM, char * msg )

 int bid = blockIdx.x;
 int tid = threadIdx.x;
 chara = ';' + bid;
 char0 = ';' + tid;
```

THE NINJA PROJECT

Pensemos un poco ahora otros aspectos

Las claves cortas no resisten un ataque offline.

Si el ataque va a durar mucho tiempo, hay que considerar ***retomar***.

Al reducir el algoritmo, el atacante gana. ¿Cómo recupero la ventaja?

Expansión de clave hasta anular optimizaciones

THE NINJ PROJECT

mas...

Este diseño responde a hallar un solo mensaje optimizando al peor caso

O para buscar distintos hashes a la vez

Trabaja en lotes, termina con el lote, no con hallar el mensaje, no está bueno para atacar un hash tras otro.

En los multithread/multiprocess, llamemoslo paralelismo externo en contraposición con SIMD, no hay coordinación tal que se interrumpan todos los hilos o procesos al hallar uno el mensaje.

Se podría adaptar fácilmente para generar, almacenar y buscar posteriormente.

Pero, ¿qué pasaría cuando se procesa 1GHashes/seg, cómo los sacás de la GPU???

THE NINJA PROJECT

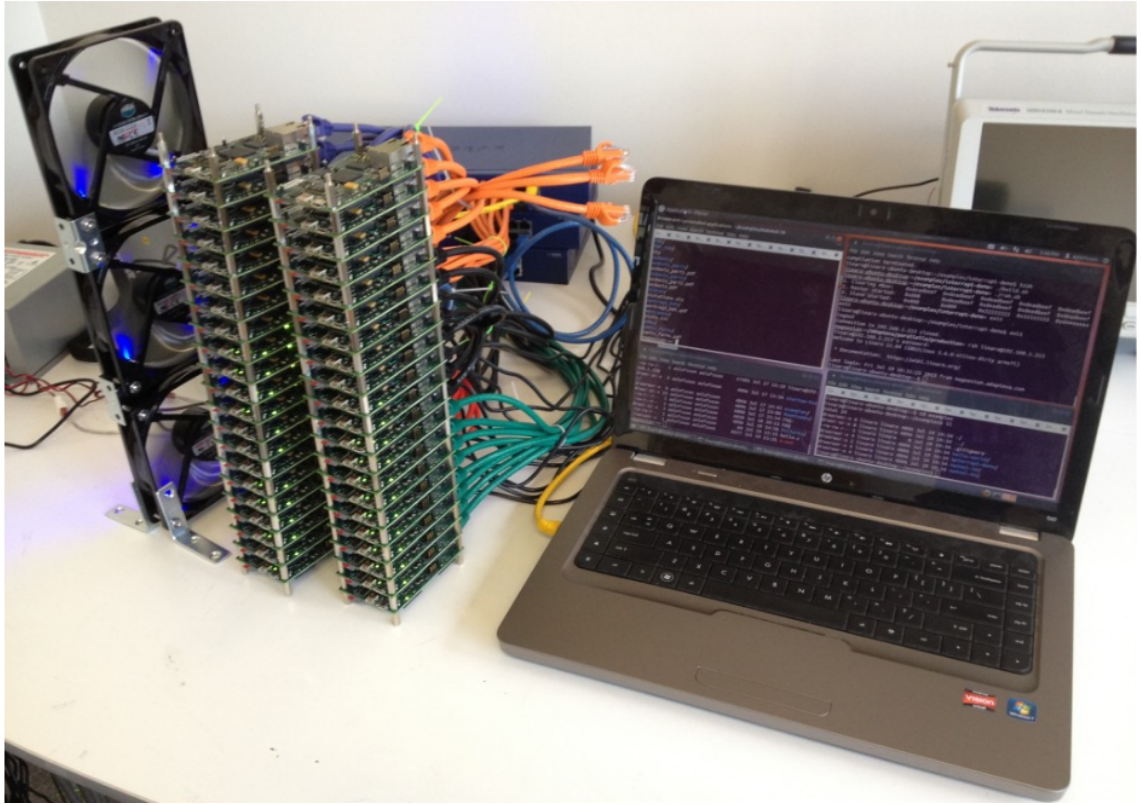
Dificultades con las cifras

- Consumo y precio de la parte
- Consumo y precio del soporte
- Año
- Equipo (parallella una persona en dos años vs nvidia/intel)
- Espacio ocupado

THE NINJA PROJECT

42 parallellas = 210 W

672 epiphany + 84 A9 cores+ 42 PL



2019-05-17
u$s 5300 +
coolers +
ps +
network

THE NINJ PROJECT

¿Las preguntas?

Algunas notas estarán desde el finde en:

https://seguridad-agile.blogspot.com/p/indice.html#md5

Y el código en:

https://github.com/cpantel/Forzando-Brutalmente-MD5

THE NINJA PROJECT