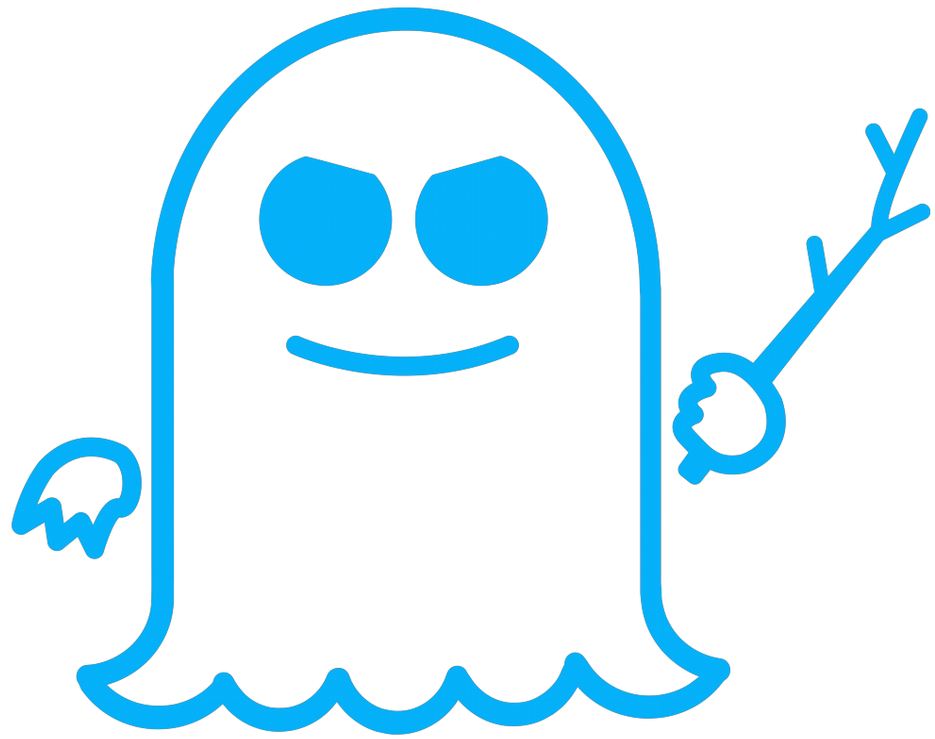
MELTDOWN



SPECTRE

- Rodolfo Azevedo
- UNICAMP

Instruction Set Architecture

The portion of the computer that is visible to the programmer or the compiler writer.

Computer Architecture: A quantitative approach

An instruction set architecture (ISA) is an abstract model of a computer. It is also referred to as architecture or computer architecture.

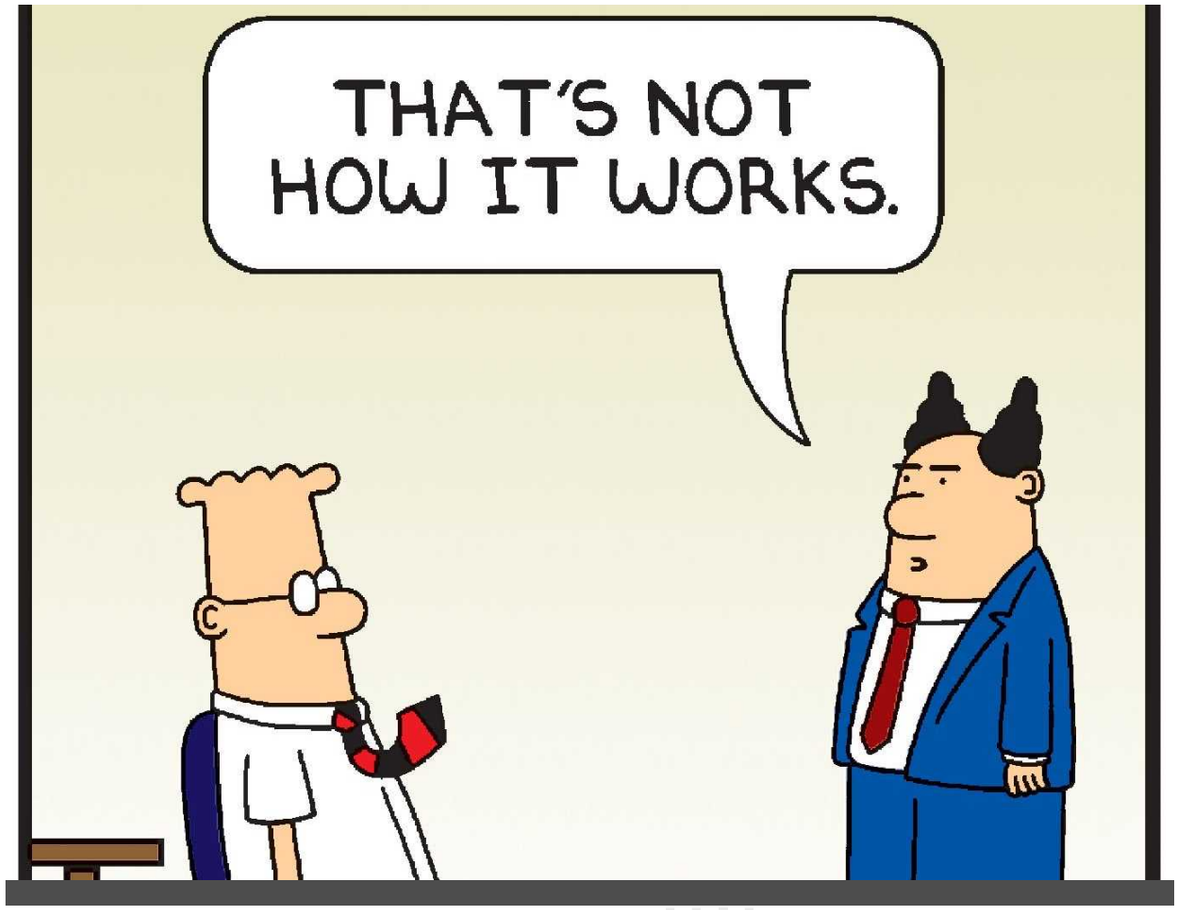
Wikipedia

Contrato entre os projetistas de hardware e software.

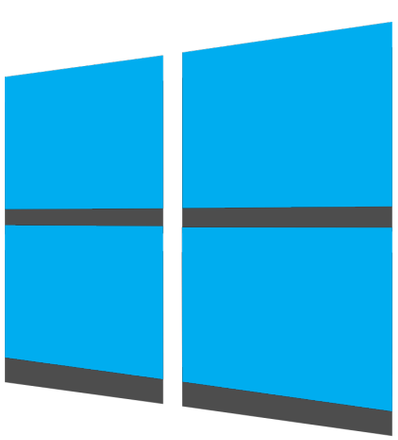
Definição de uma linha

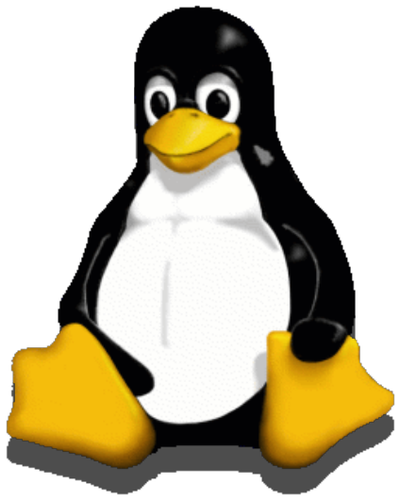
Que tipo de contrato é esse?

- A empresa fornece a especificação precisa e você escreve o software!



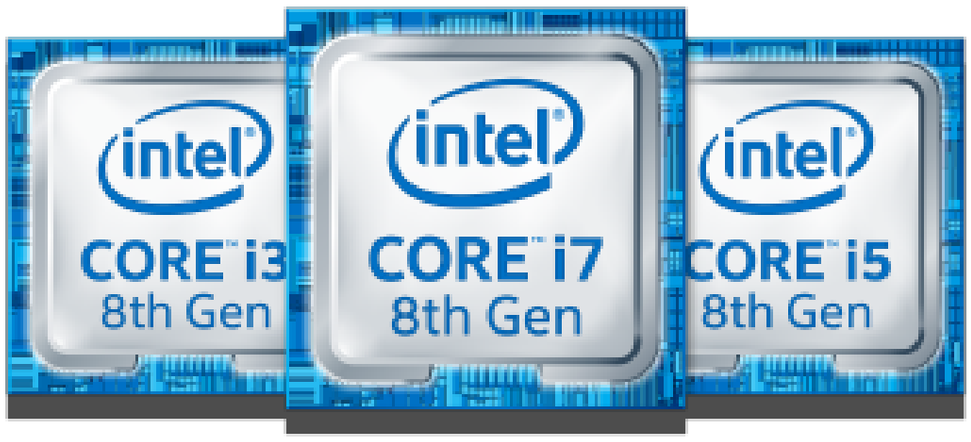
Isto é importante?



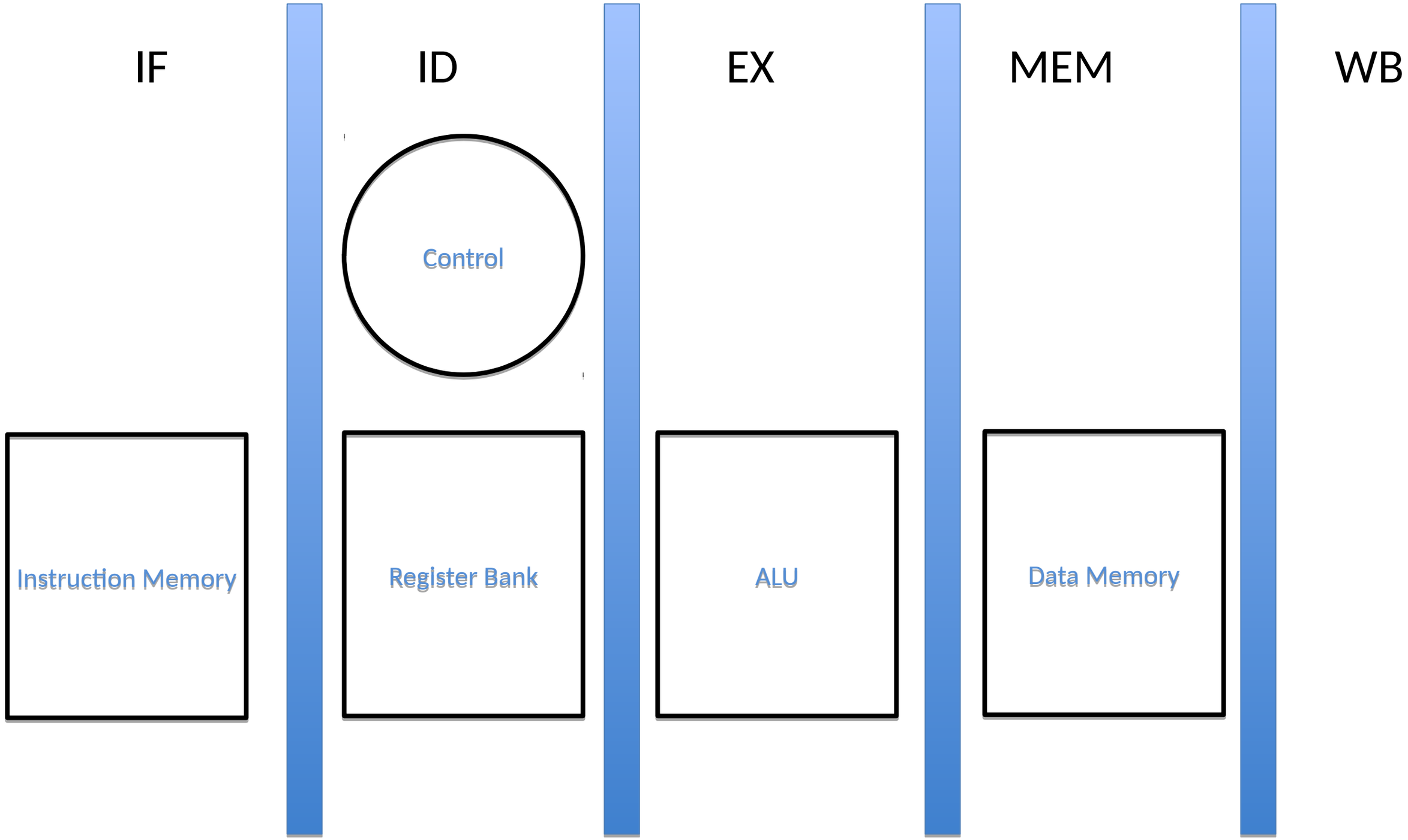


OS X

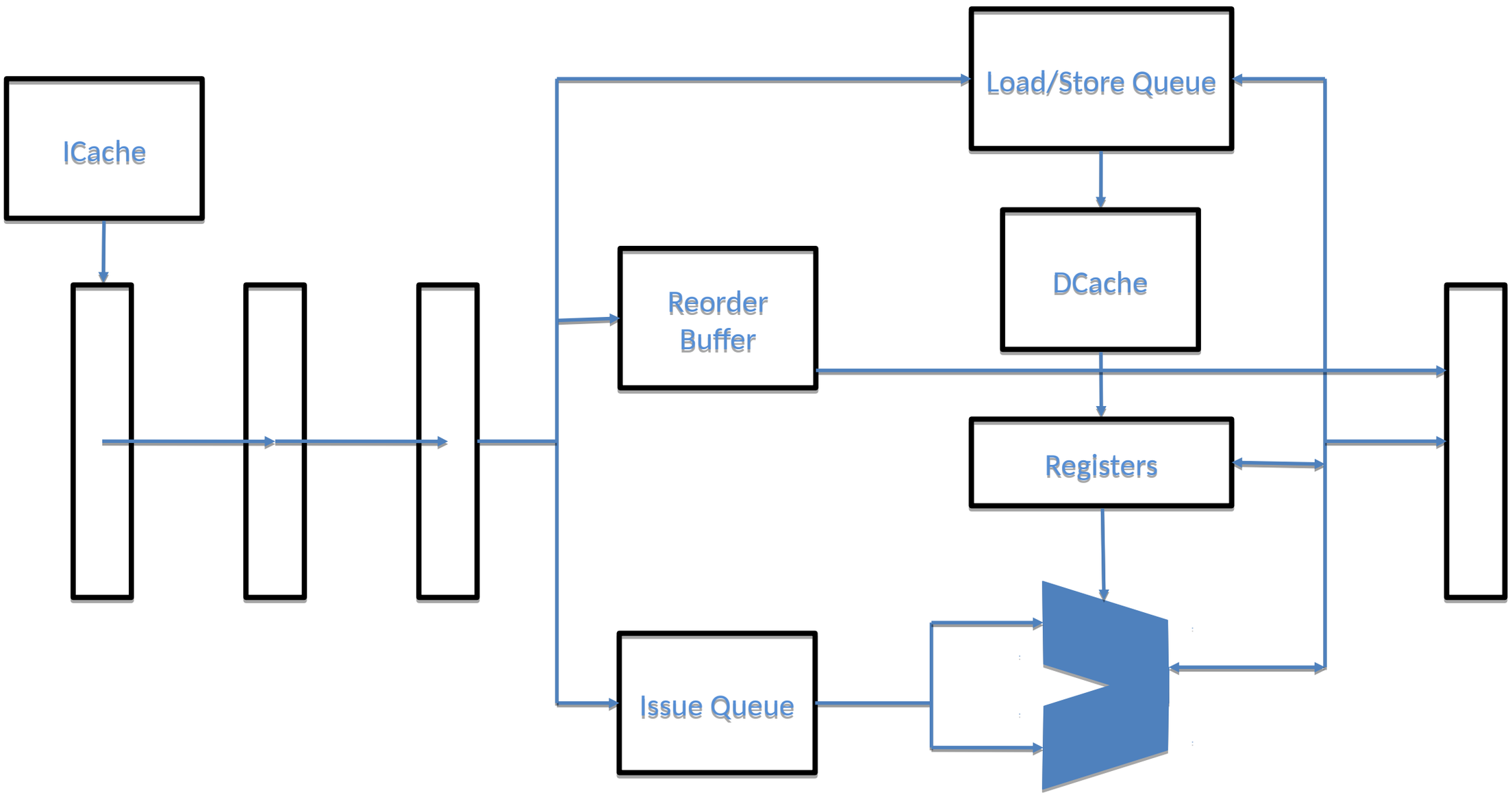
x86 ISA



Visão altíssimo nível de um pipeline escalar



Visão alto nível de um superescalar



Temporização

Contador de tempo do processador

- **rdtsc**: lê o *time stamp clock*
  - *Pode ser reordenada no código*

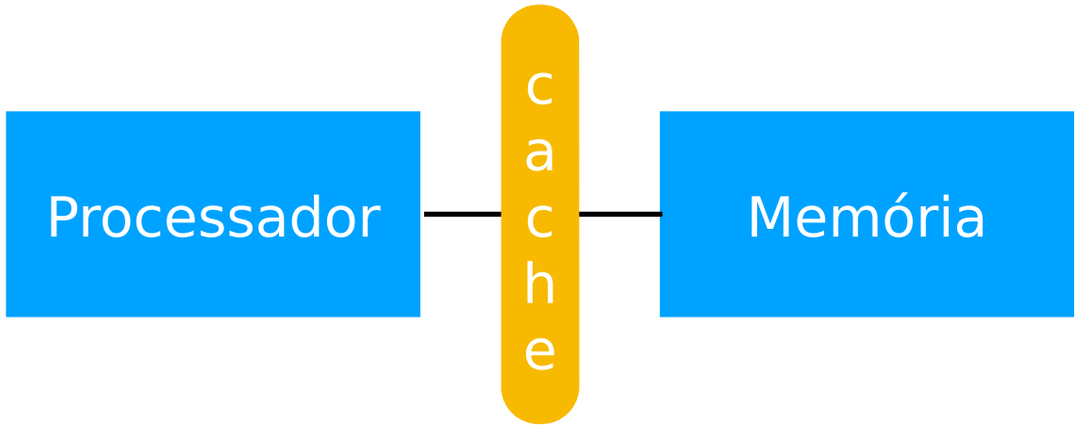
```
__asm__ __volatile__ ("RDTSC\n\t"
        "mov %%edx, %0\n\t”
        "mov %%eax, %1\n\t": "=r" (cycles_high),
                             "=r" (cycles_low)::
                          "%rax", "rbx", "rcx", "rdx"
```

- **rdtsp**: lê o *time stamp clock e processor id*
  - *Garante que todas as instruções que iniciaram antes terminarão antes*

Caches

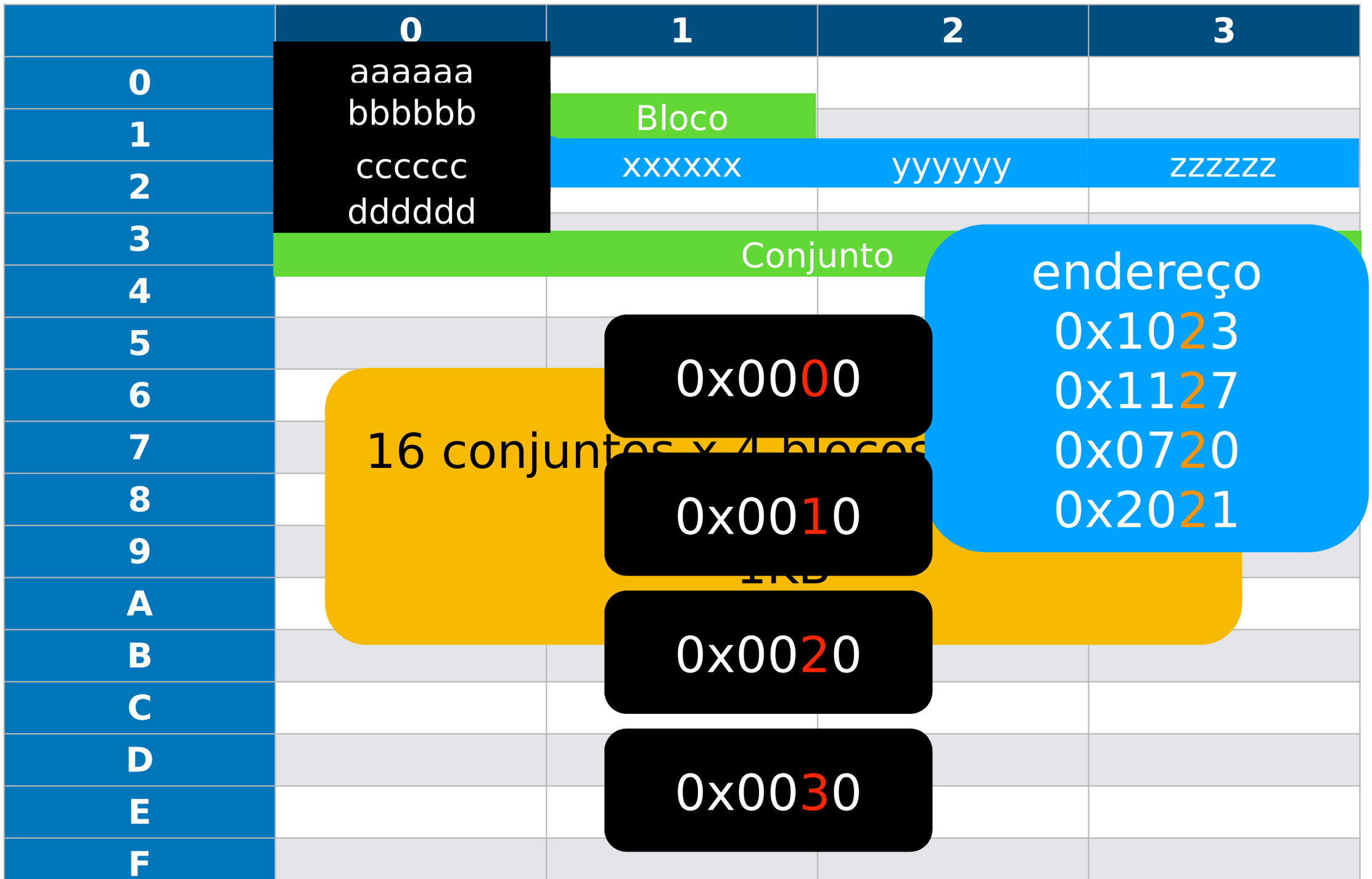
Caches

- DRAM são lentas
- A hierarquia de caches guarda cópia dos dados recentemente utilizados acelerando acessos posteriores
- Acesso transparente ao usuário



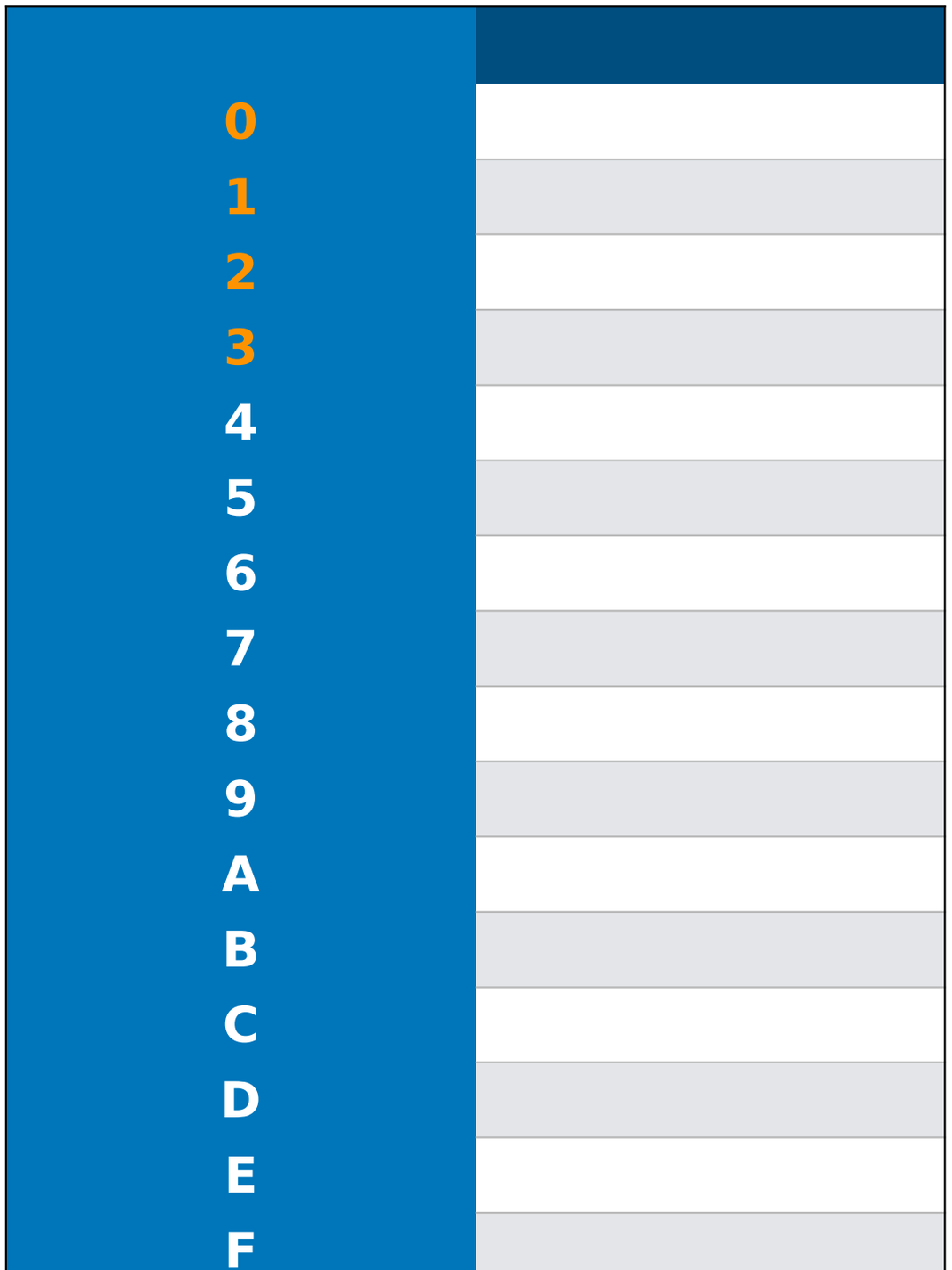
—> Ler posição 1000 Miss
<— valor 10
—> Ler posição 2000 Miss
<— valor 20
—> Ler posição 1000 Hit
<— valor 10

Por dentro da cache



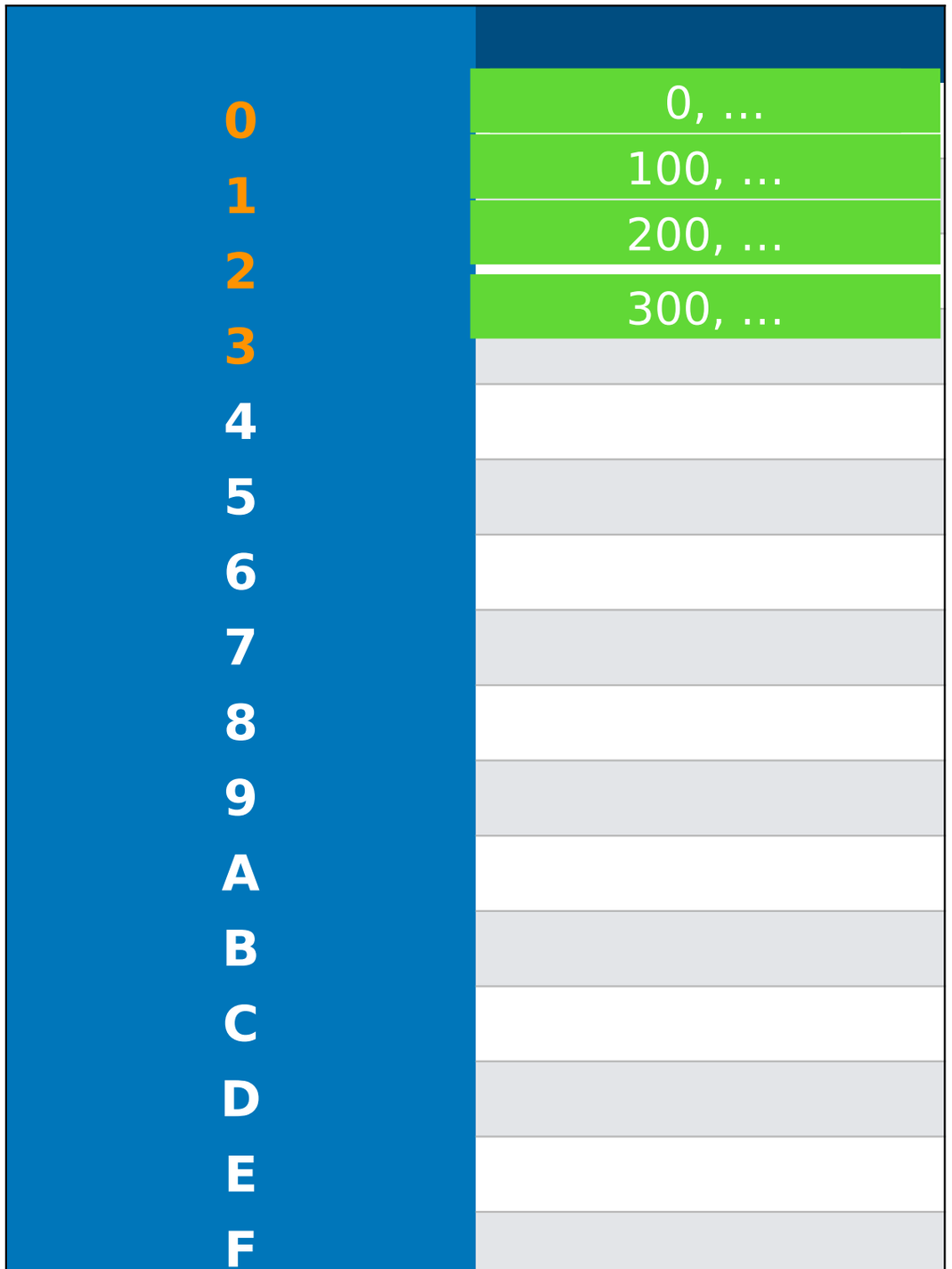
Exemplo

- 7 voluntários
  - 4 células da cache
  - 1 controlador
    - 1 cronômetro
    - 1 memória
- Lembrete sobre o endereço da cache: 0x00N0



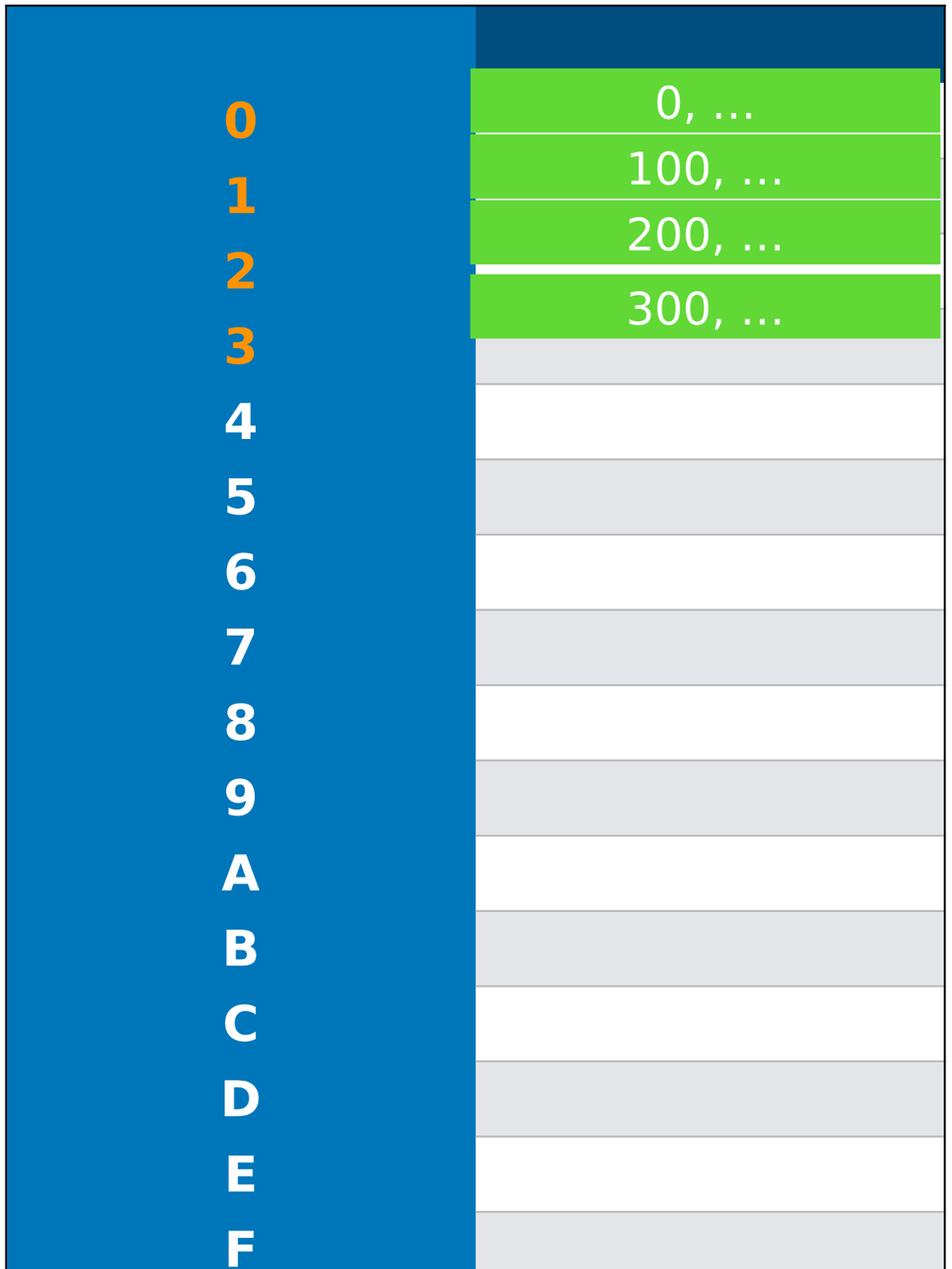
Treinamento

- 0x1000
- 0x0000
- 0x0010
- 0x0020
- 0x0030
- 0x0000



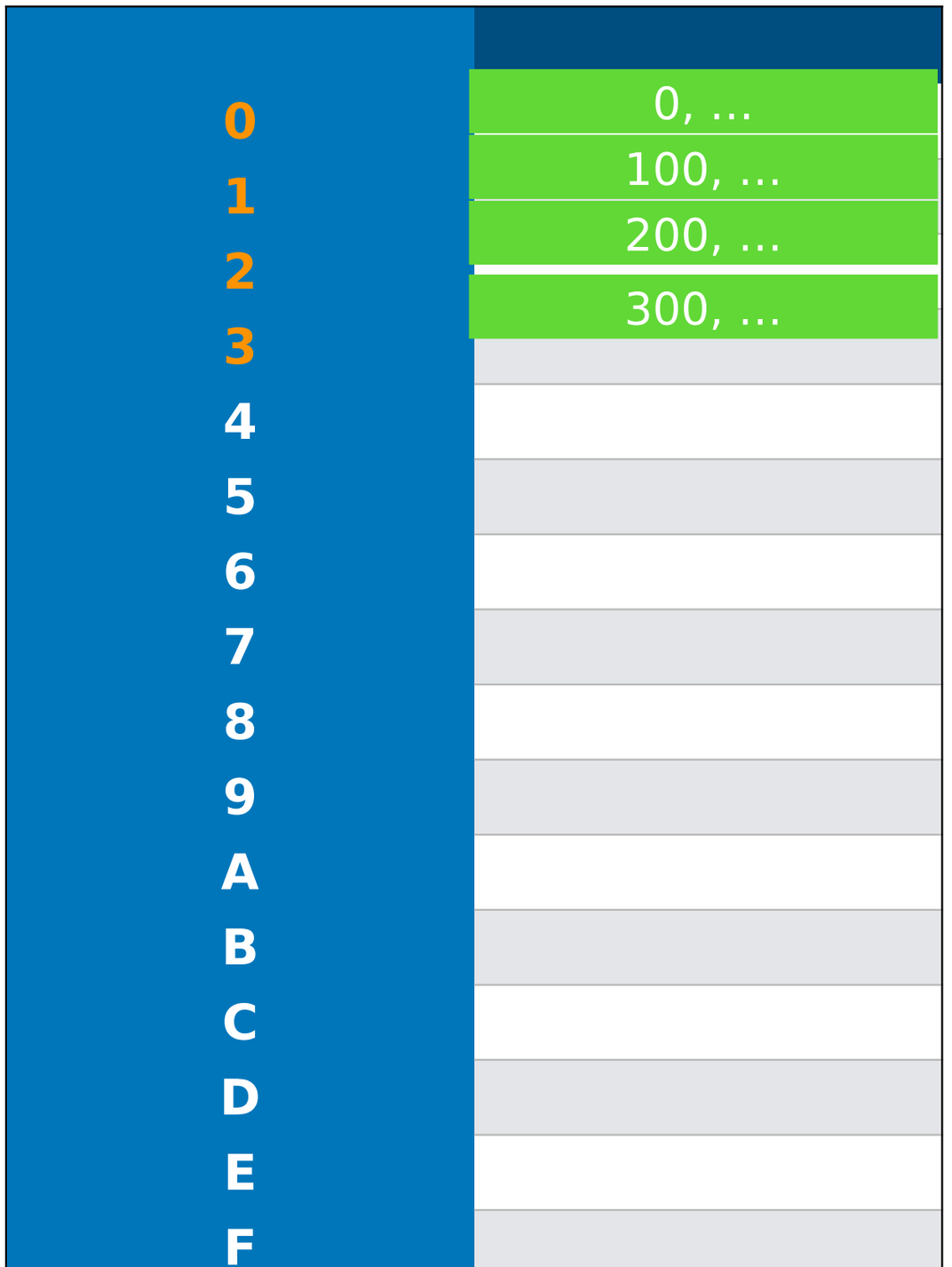
Uma leitura nova

- ??????



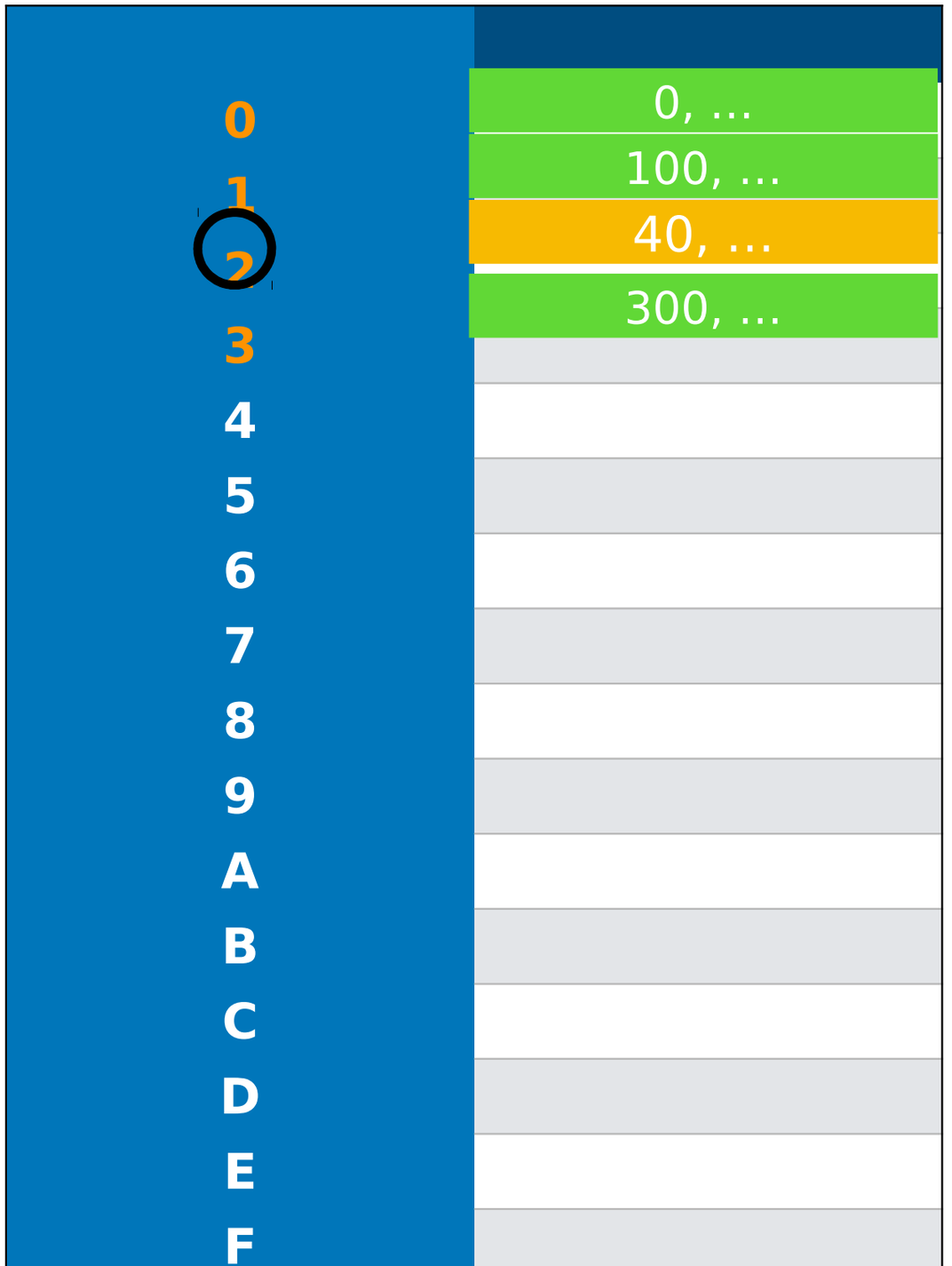
Vamos descobrir algo sobre a leitura

- 0x0000
- 0x0010
- 0x0020
- 0x0030



Conclusões

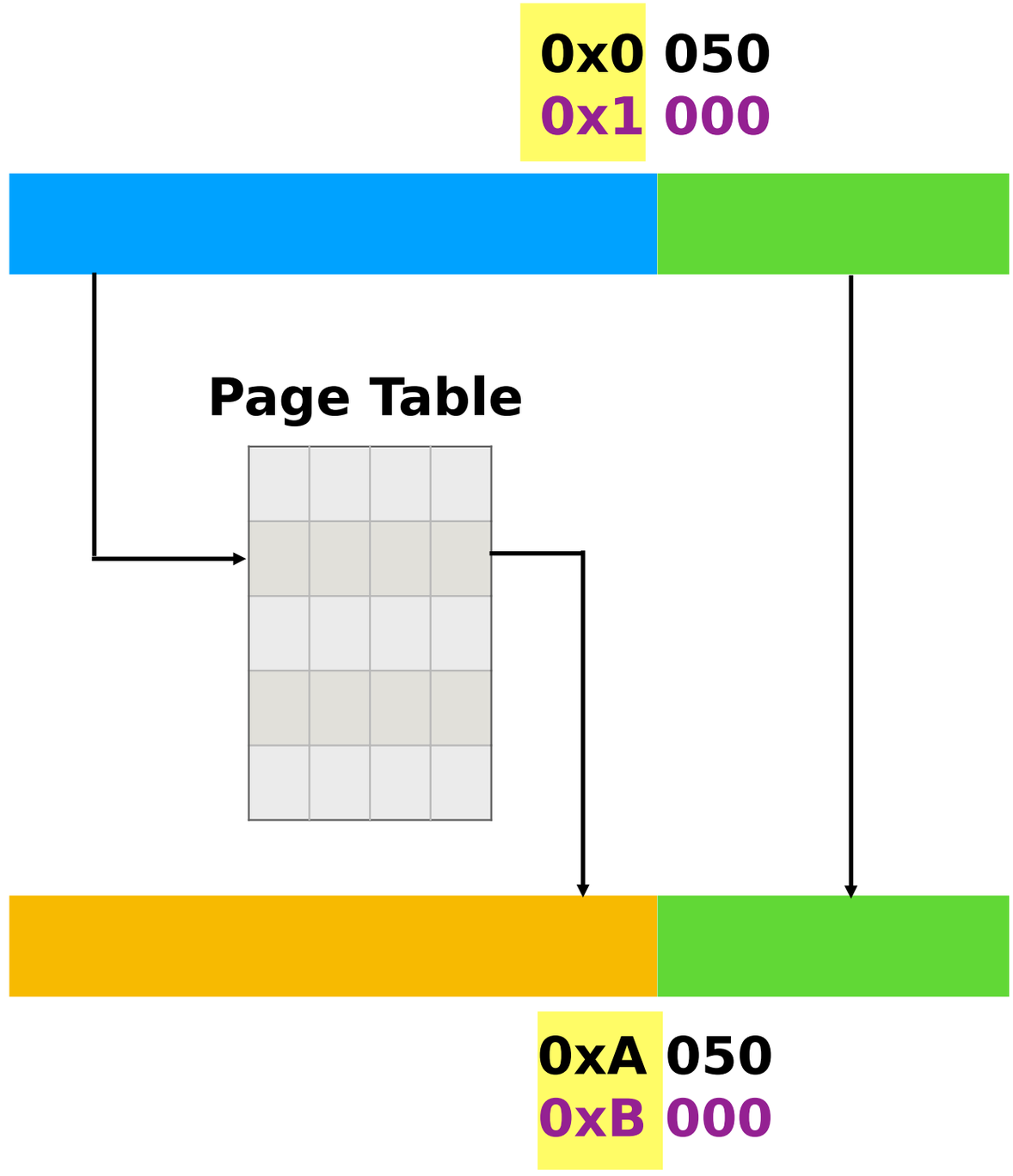
- Pode-se converter dados em endereços e testar várias posições de memória



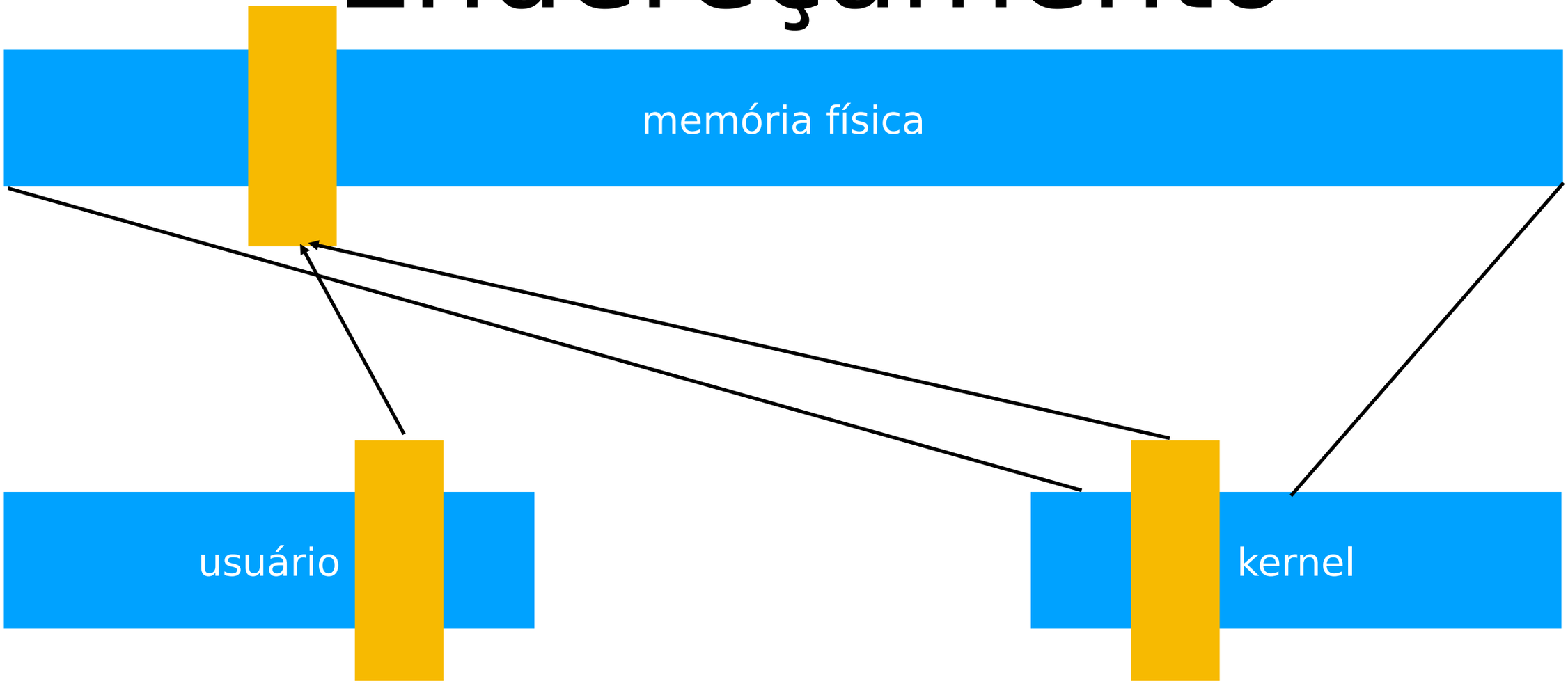
Memória Virtual

Memória Virtual

- Converte endereços virtuais em físicos (reais)
- Implementa mecanismos de proteção de memória
- Page Table
  - Hierárquica
  - Bits de proteção
- TLB
  - Acesso rápido aos dados da Page Table



Espaço de Endereçamento



inux e MacOS: todo o espaço de endereçamento é mapeado no kernel
Windows trabalha com pools de memória

Exemplo bem simples

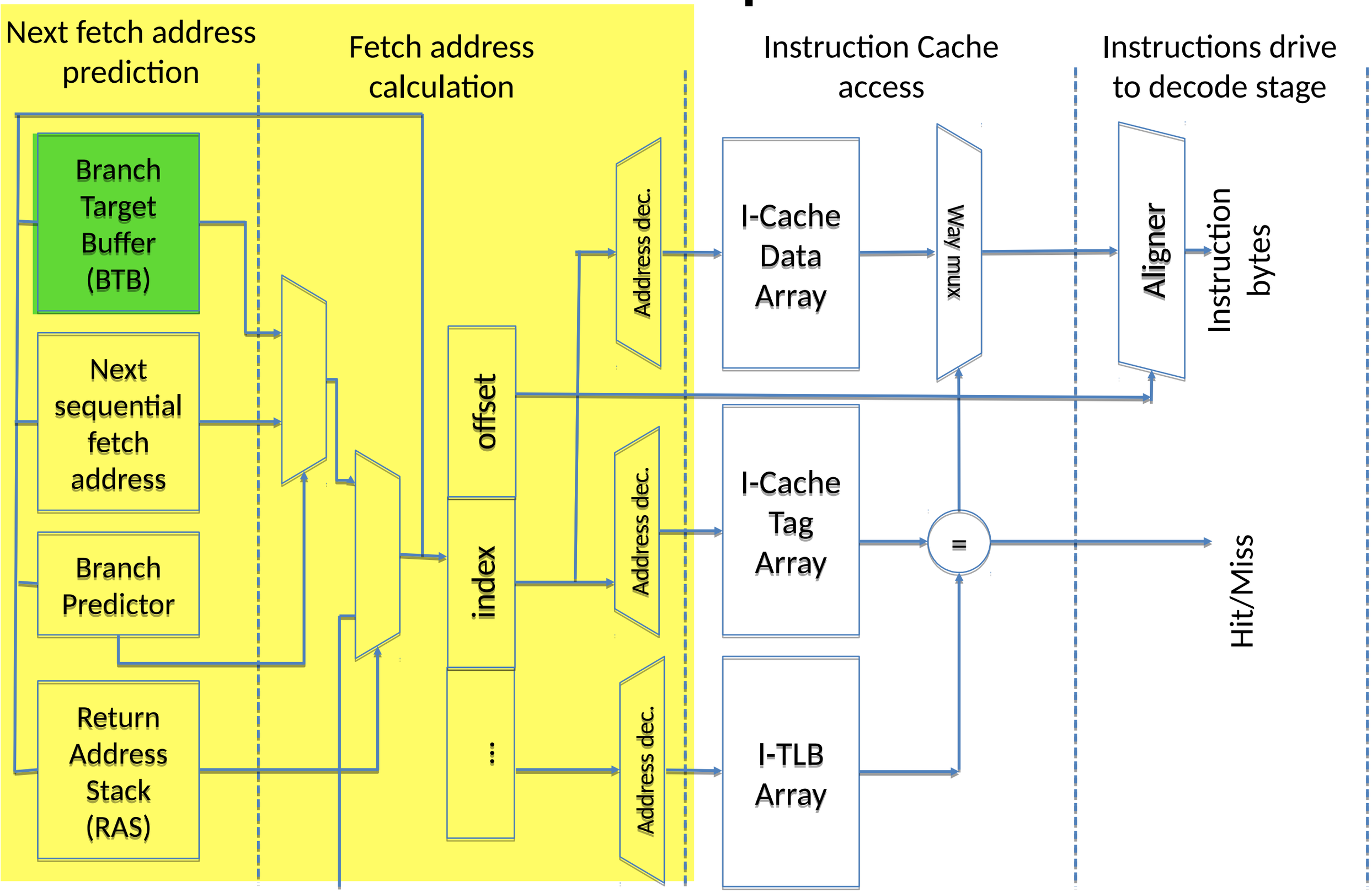
- Endereços começados com 0: usuário
- Endereços começados com outros dígitos: supervisor
- Voluntário?
- 0x0000
- 0x1000
- 0x2000

KASLR

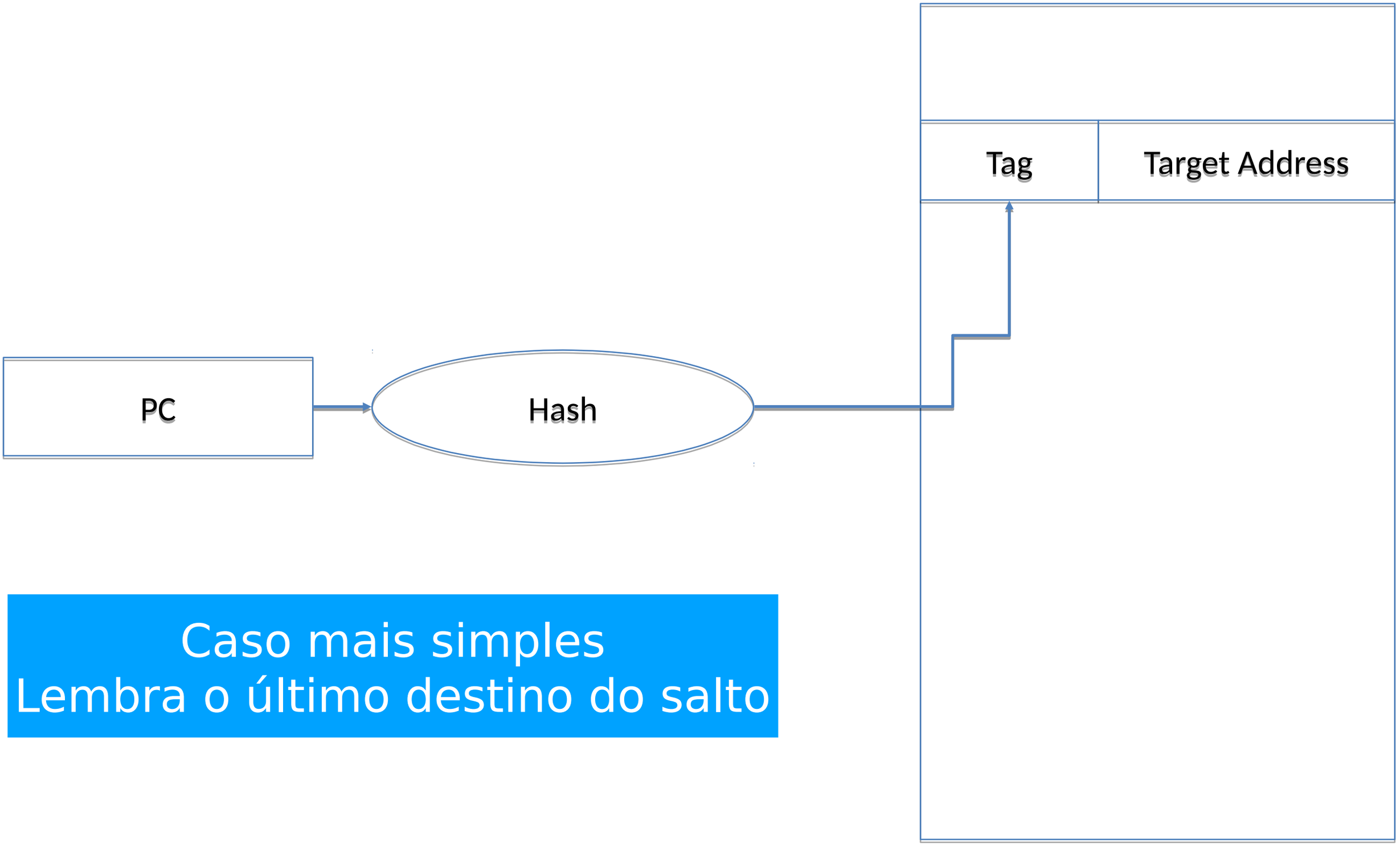
- Kernel Address Space Layout Randomization
- Endereços onde o kernel é carregado são aleatorizados toda vez que o sistema inicia
- Melhora a segurança contra ataques, mas ainda é possível descobrir onde o kernel está carregado
- Utiliza blocos de 8GB para carga

Branch Prediction

Fetch Pipeline



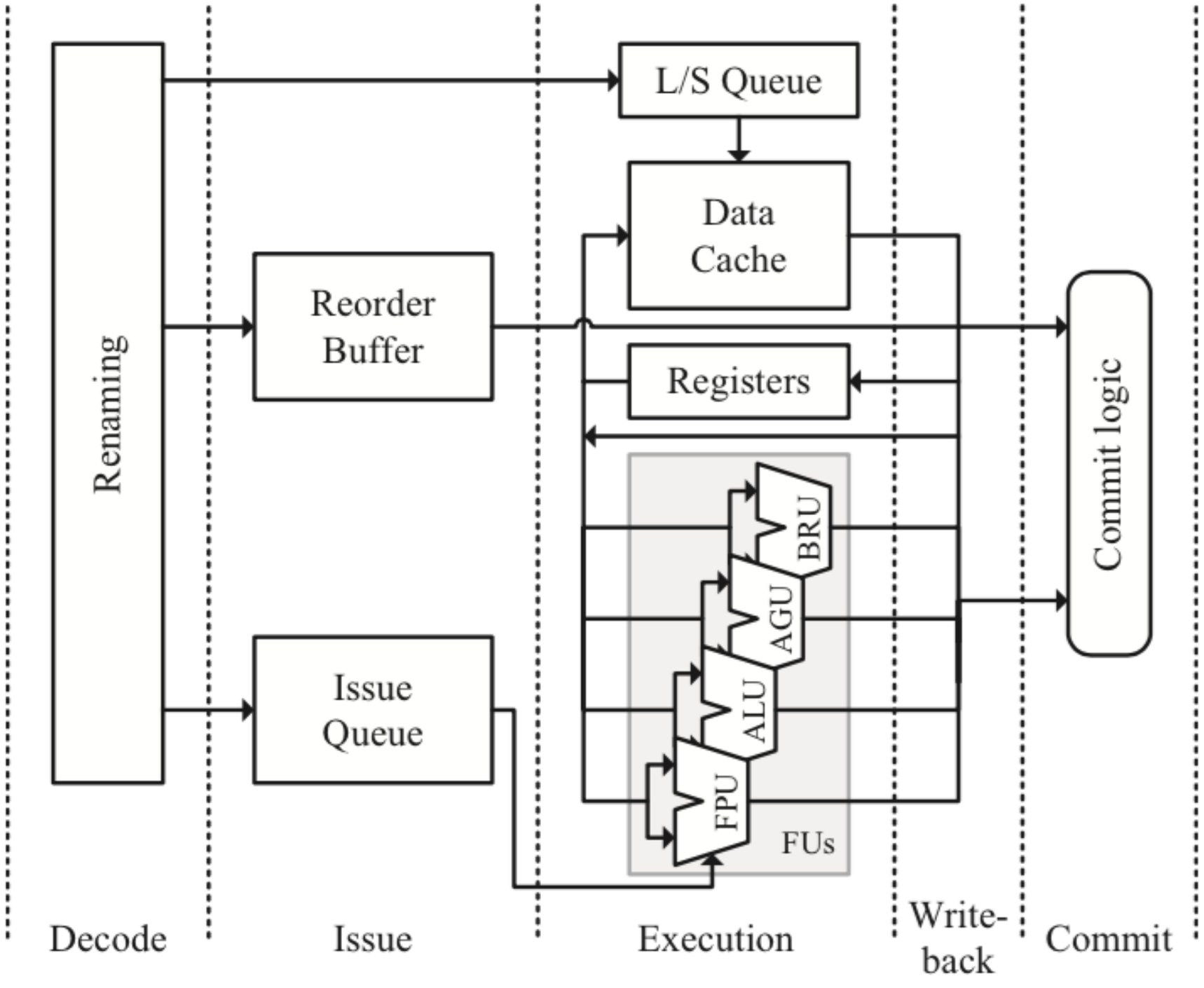
Branch Target Buffer



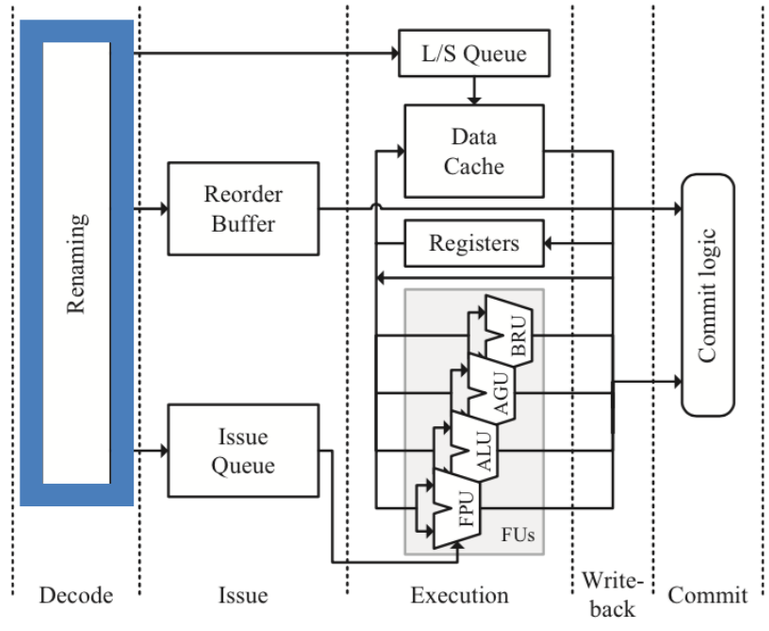
Caso mais simples
Lembra o último destino do salto

Execução fora de ordem

Execução


L/S Queue
Data Cache
Reorder Buffer
Renaming
Registers
BRU
AGU
ALU
FPU
FUs
Commit logic
Issue Queue
Decode
Issue
Execution
Write-back
Commit

Renaming



- Utiliza mais registradores internos que a quantidade disponível na ISA

- Remove dependências de nome

```
p1 = r2 + r3
...
r4 = p1 + r5
```

Data dependence
Read after write

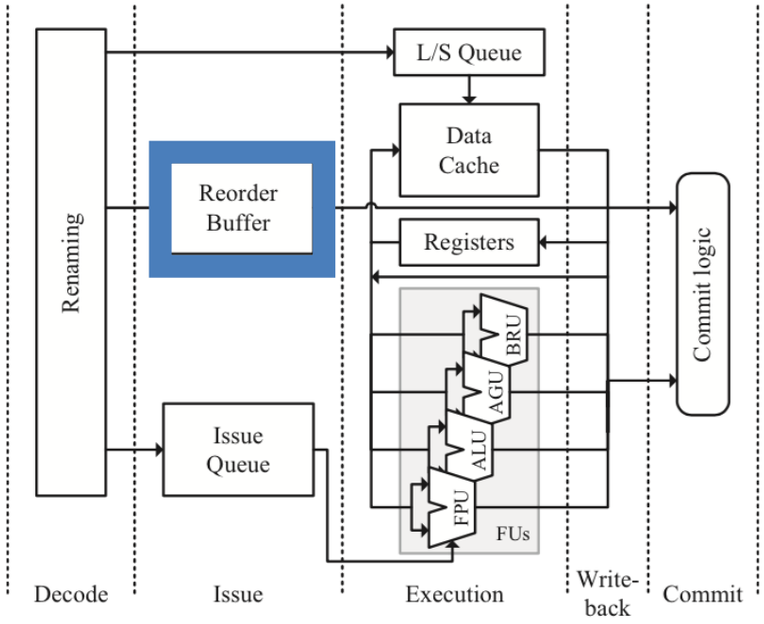
```
p1 = r2 + r3
...
p2 = r4 + r5
```

Name dependence
Write after write

```
r1 = p1 + r3
...
p2 = r4 + r5
```

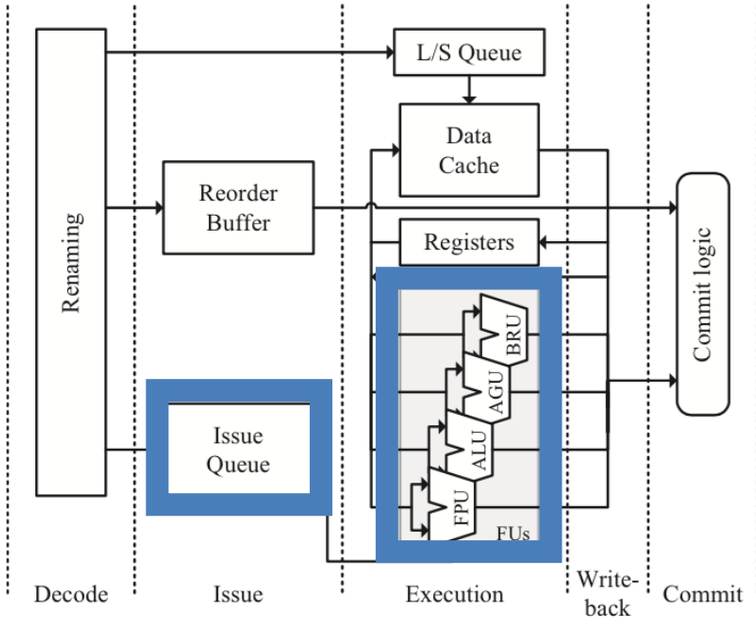
Name dependence
Write after read

Reorder buffer



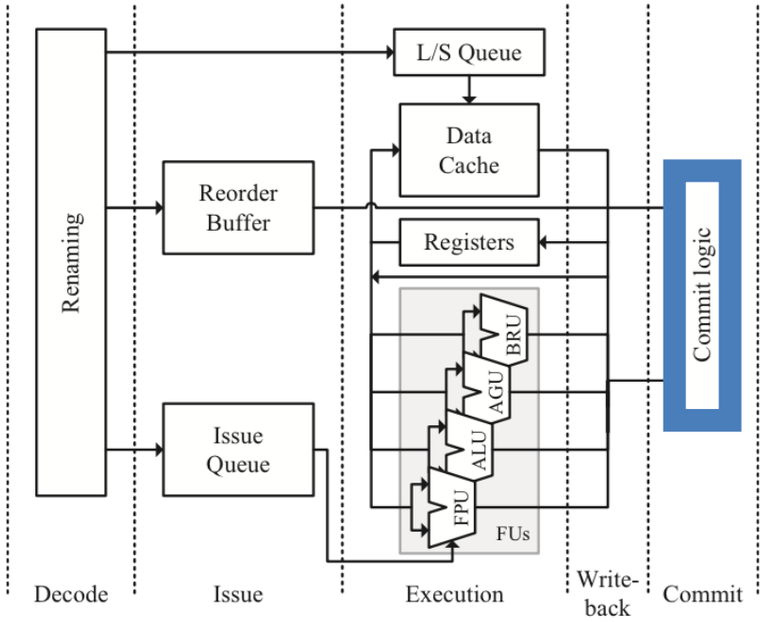
- Guarda as instruções na ordem de execução do código
- Guarda bits extras sobre o estado da execução das instruções
  - Exception bit
- A instrução será efetivada no estágio de commit

Issue Queue



- Envia instruções para as unidades funcionais
- As instruções podem ser executadas fora de ordem
  - Desde que não haja dependência não resolvida ainda

Commit



- Efetiva as instruções em ordem
- Gera as exceções

Será que funciona?

Atores

- Voluntários da cache
- Estágio de fetch
- 2 para execute
- 1 para gerenciamento de memória virtual
- 1 para commit

Instruções em papel

O que houve

- Memória Virtual
  - Marca com * quando não há permissão para executar
- Commit
  - Para a execução quando recebe uma instrução com *
- Cache
  - Comportamento conhecido

Meltdown

- Instrução 1
  - Leia posição de memória 0x2020
- Instrução 2
  - Pegue o resultado da instrução anterior e acesse a memória

Código

```
retry:

mov al, byte [rcx]

shl rax, 0xc

jz retry

mov rbx, word [rbx + rax]
```

Leitura não permitida

prepara o endereço

acessa a memória

Requisitos extras

- Acesso a endereço inválido gera exceção
- Programas de usuário podem capturar esta exceção e não fazer nada
- Pode-se usar instruções de memória transacional para evitar as exceções. As transações serão abortadas
- Pode ser executado por Javascript, no seu browser, apontando para uma página maliciosa

Spectre

```
if (x < array1_size)
    y = array2[array1[x] * 256];
```

Falso mas previsto como verdadeiro

Comentários

- Totalmente no nível de usuário
- Processadores multithread compartilham estruturas de branch prediction
- É possível ler dados do seu browser via Javascript
  - Poluir o branch predictor não é tão simples mas é possível

Diferenças

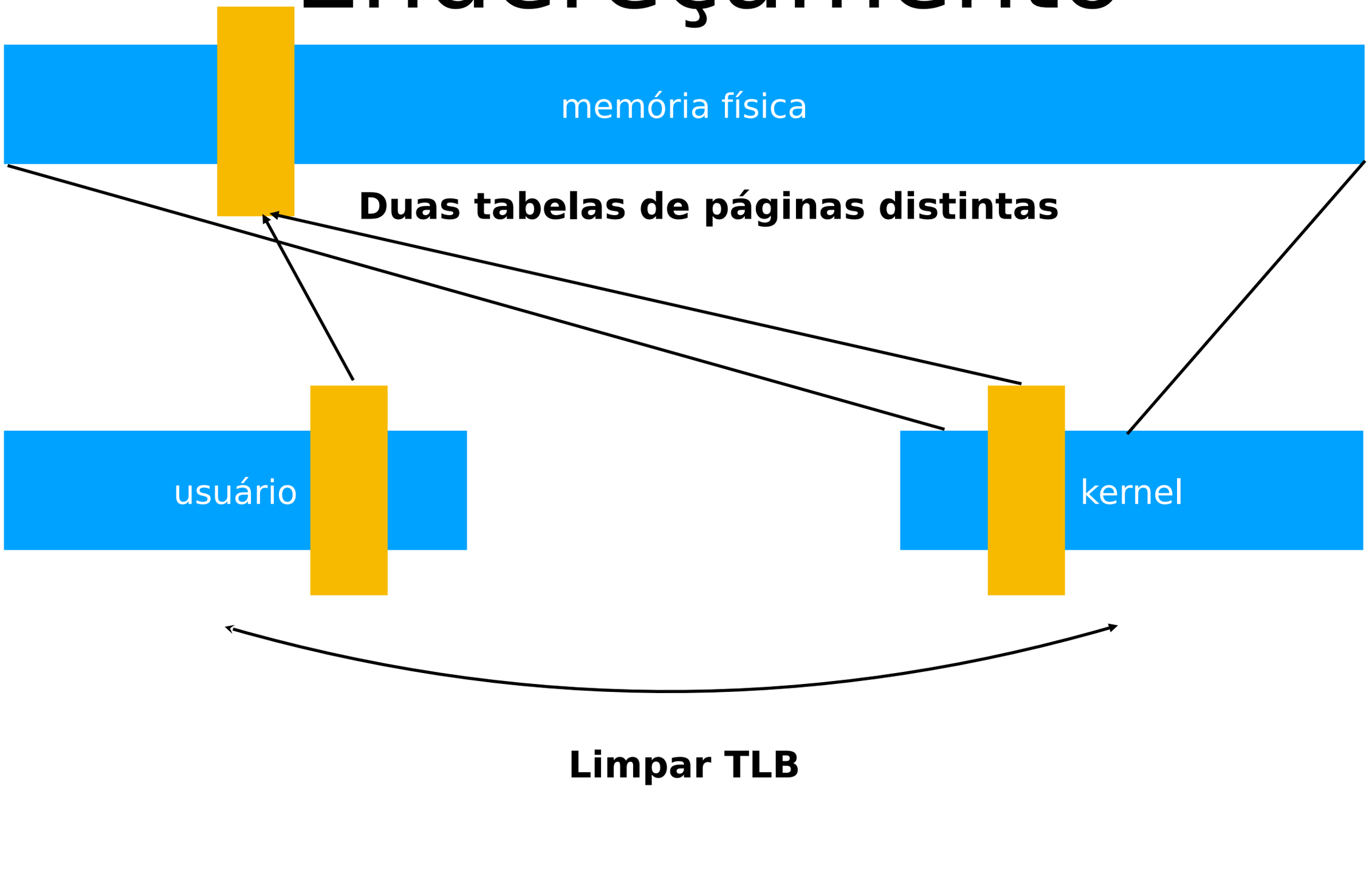
	Meltdown	Spectre
Architecture	Intel, Apple	Intel, Apple, ARM, AMD
Entry	Must have code execution on the system	Must have code execution on the system
Method	Intel Privilege Escalation + Speculative Execution	Branch prediction + Speculative Execution
Impact	Read kernel memory from user space	Read contents of memory from other users' running programs
Action	Software patching	Software patching (more nuanced)

https://danielmiessler.com/blog/simple-explanation-difference-meltdown-spectre/

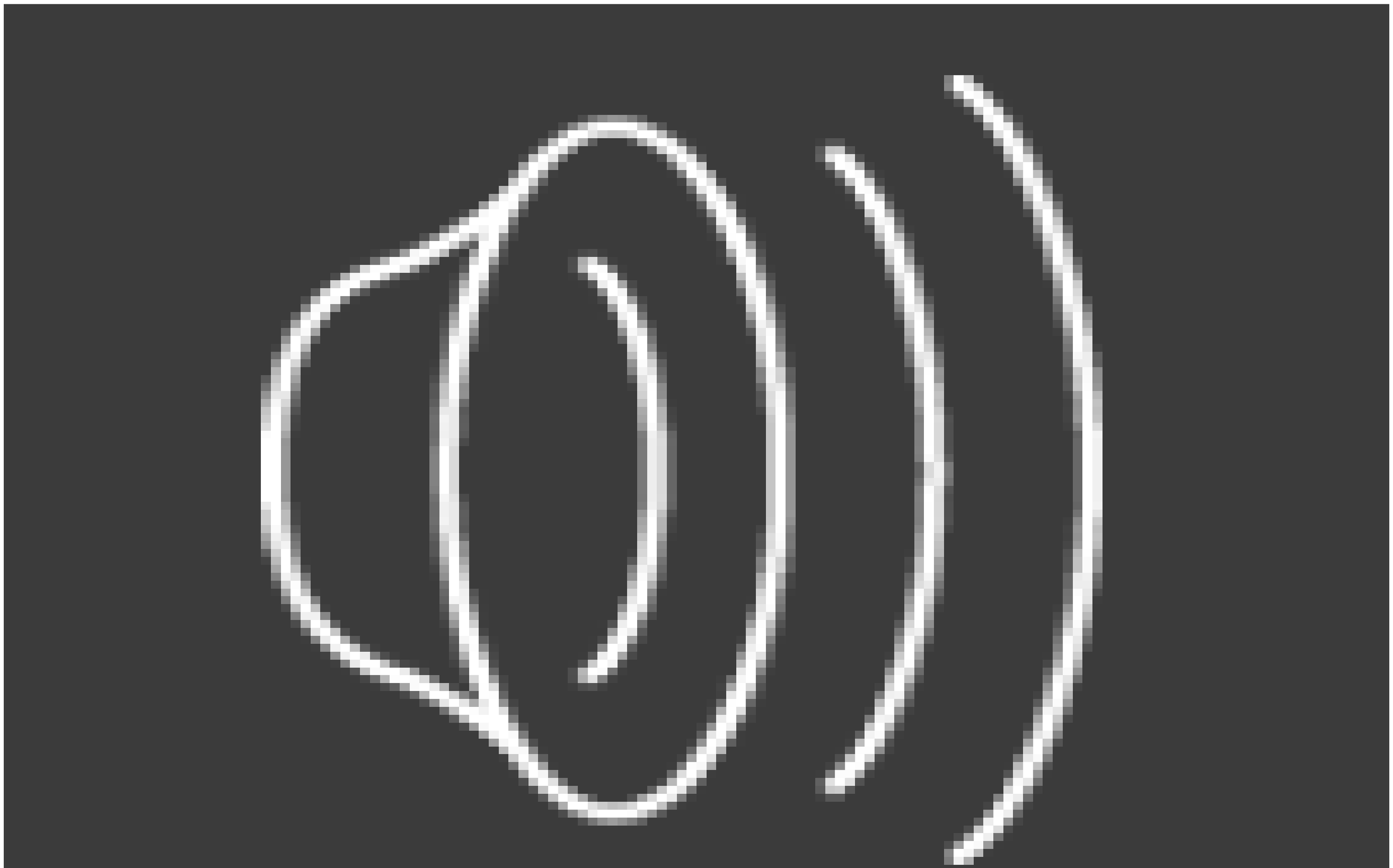
Patches

- Meltdown: Separar o espaço de endereçamento do kernel das aplicações
  - KAISER Patch
  - Mais TLB flushes durante a execução de programas
- Spectre: With microcode updates, Intel has enabled three new features in its processors to control how branch prediction is handled.
  - IBRS ("indirect branch restricted speculation") protects the kernel from branch prediction entries created by user mode applications;
  - STIBP ("single thread indirect branch predictors") prevents one hyperthread on a core from using branch prediction entries created by the other thread on the core;
  - IBPB ("indirect branch prediction barrier") provides a way to reset the branch predictor and clear its state.

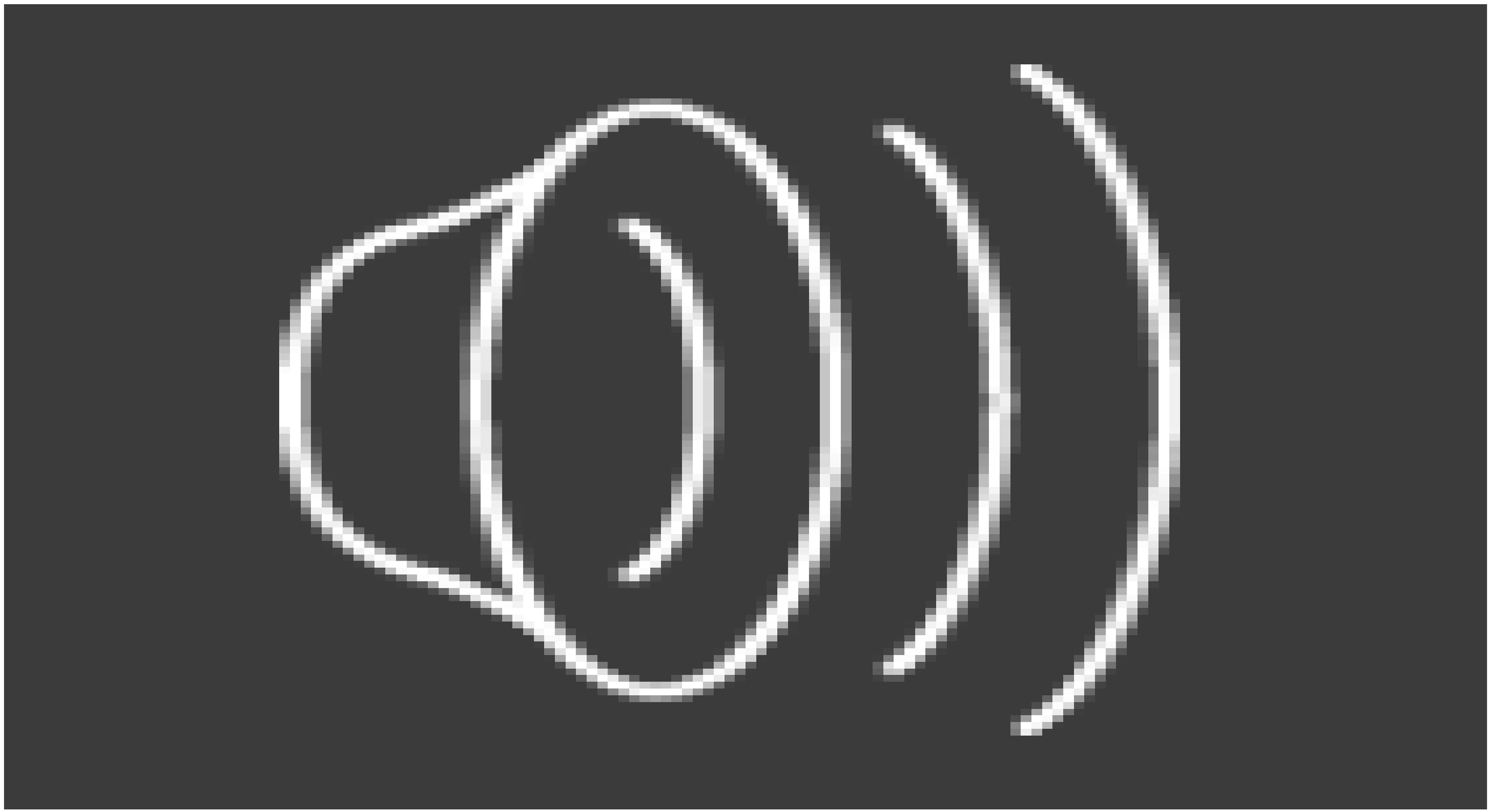
Espaço de Endereçamento



Meltdown in action
memory dump



Meltdown in action password



Conclusões

- Nova era de timing attacks em grande escala
- Repensar a especificação da arquitetura (ISA)
  - Incluir alguns requisitos de temporização?
  - Especulação?
  - Maior separação dos estados internos dos processos

Referências

- http://meltdownattack.com

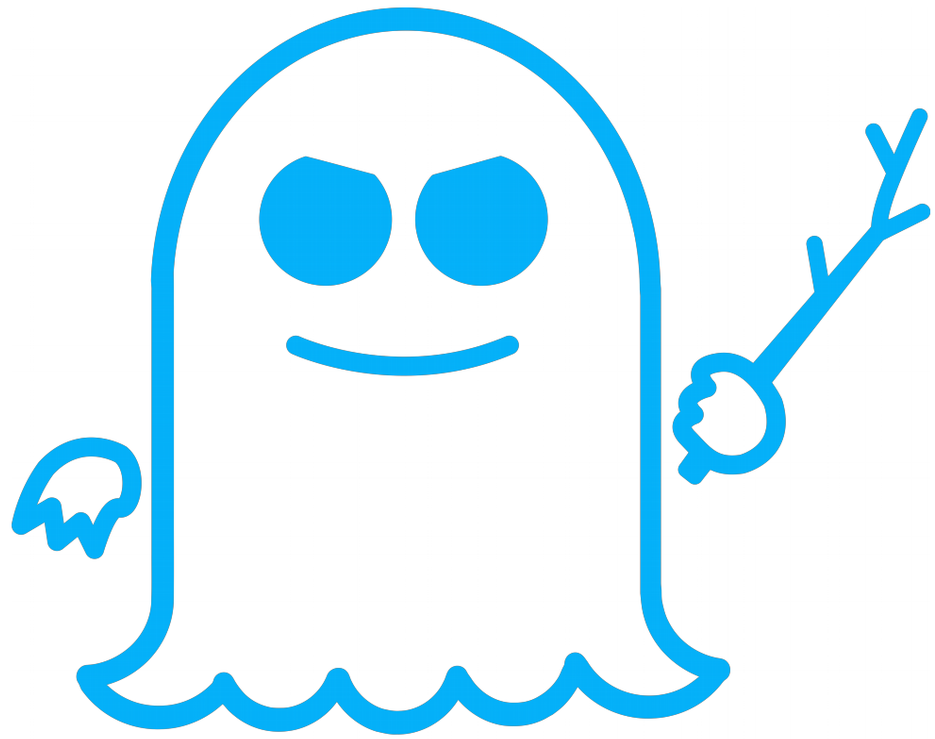
- Meltdown: https://meltdownattack.com/meltdown.pdf

- Spectre: https://spectreattack.com/spectre.pdf

- Exemplos no github: https://github.com/IAIK/meltdown

- On the Meltdown and Spectre Design Flaws. Mark Hill. http://research.cs.wisc.edu/multifacet/papers/hill_mark_wisconsin_meltdown_spectre.pdf

MELTDOWN



SPECTRE

- Rodolfo Azevedo
- UNICAMP