Arquitetura de Computadores

MiEI – 2016/17

DI-FCT/UNL

Aula 22

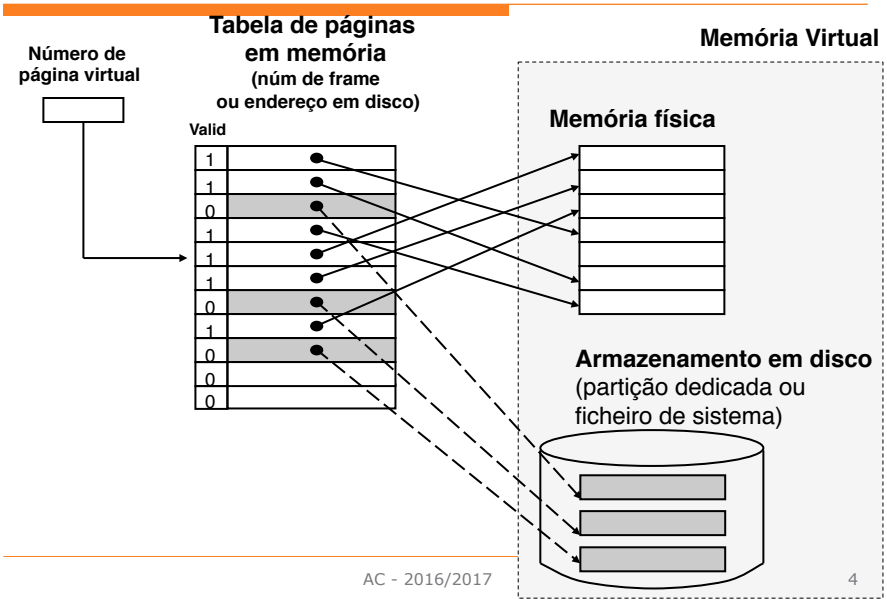
Memória virtual com paginação

- Memória virtual (VM) – separação da memória lógica (virtual) da memória física
  - MV definida pelo espaço de endereços lógicos (ou virtuais)
  - Só parte da imagem do processo precisa de estar em memória
  - É possível ter memória real < soma das imagens de todos os processos
  - Se espaço de endereços virtuais > memória real num computador, um único processo pode "usar" uma memória virtual > memória real
- É completamente transparente para programador/ programa
  - O SO fica responsável por oferecer a visão de memória virtual
  - Usa o disco para “estender” a memória real, gerindo a memória real como uma **cache** para todas as imagens dos processos

Memória virtual com paginação a pedido

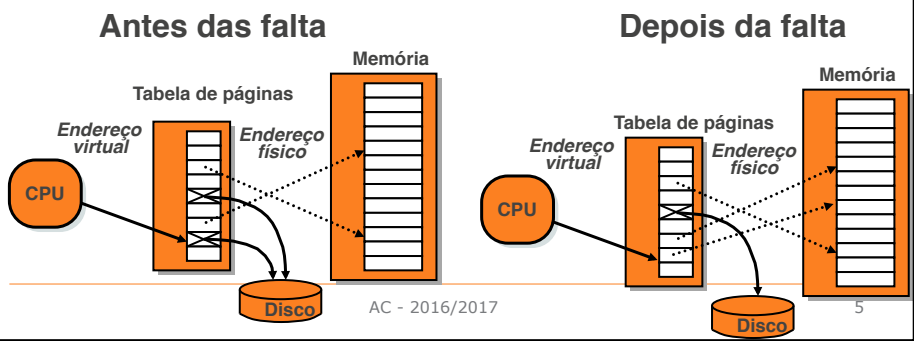
- MV pode ser suportada através de **paginação a pedido**:
  - Cada página virtual só é mapeada em memória real (ou seja, só lhe é atribuído um frame) quando é referenciada, i.e. quando um endereço virtual emitido pelo CPU corresponde a esse número de página
  - No caso do código e dados inicializados, o respectivo conteúdo é também lido do respectivo ficheiro executável
  - Todas as páginas da imagem do processo podem, se necessário, ser guardadas em disco, chamado **disco de paginação** ou disco de **swap**
  - As páginas em *swap* só voltam para memória central quando são novamente referenciadas

Tabela de páginas e memória virtual



Falta de página

- Tabela de páginas indica que o conteúdo referenciado pelo endereço virtual não está em RAM
- É invocado um procedimento de exceção para trazer os dados para memória (**swap in**)
  - o SO tem controlo completo sobre a localização das páginas

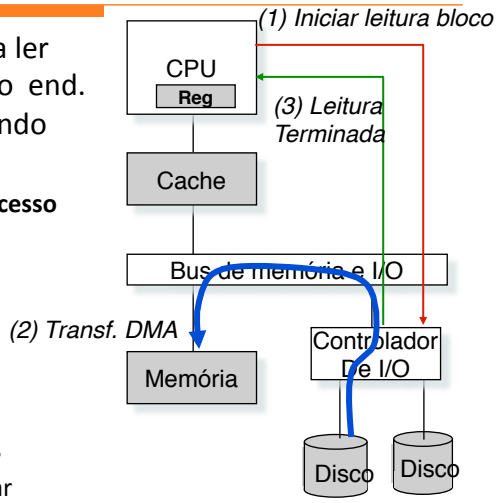


Falta de página (Page Fault)

- Uma referência a uma página que não está em memória provoca uma exceção (fault) e o SO intervém:
  - Referência inválida ⇒ aborta o programa.
  - válida mas falta a página => trata a falta de página
- Trata falta de página:
  - Obtém uma "frame" vazia se houver. Se não, tem de libertar uma.
  - Carrega a página nessa "frame".
  - Actualiza a tabela de páginas
    - Tab_paginas[P].frame = F
    - Tab_paginas[P].V = 1.
- Retoma a instrução que provocou a exceção

Atendimento da falta de página

- CPU pede ao controlador para ler um bloco de dim. P a partir do end. X e escrever na RAM começando no end. Y
  - **O CPU pode executar outro processo**
- Ocorre a leitura
  - Direct Memory Access (DMA)
  - sob controlo do I/O controller
- I / O Controller assinala o fim
  - Interrupção
  - SO actualiza a tabeça de páginas
  - Processo anterior pode continuar



Obter frame vazia

- Se não há uma frame vazia, é invocado um algoritmo de substituição de páginas para escolher uma "vitima"
  - Substituição global: a frame escolhida como vítima pode pertence a qualquer processo
  - Substituição local: a frame é escolhida entre as que já estão atribuídas ao processo.
- Se a frame escolhida foi escrita (*dirty*) é preciso escrever o seu conteúdo no disco de paginação na zona correspondente

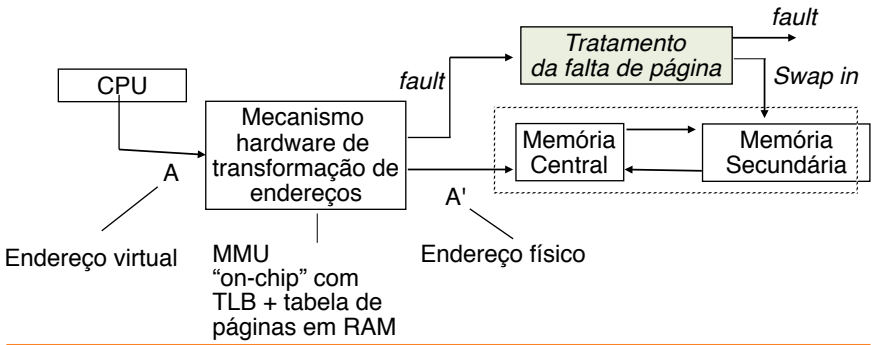
Paginação a pedido

- A página só é trazido para memória quando é referenciada (quando é pedida).
  - Menos I/O
  - Menos RAM utilizada
  - Tempo de resposta menor
  - Mais programas em RAM
- Página é necessária ⇒ referencia-se; se não está em memória (bit Val = 0), o MMU gera uma exceção (fault). O SO avalia qual dos casos seguintes se trata:
  - Referência inválida ⇒ abortar a execução
  - Não-está-em-memória ⇒ é preciso trazer a página virtual para RAM
    - Pode provocar swap out de uma página "vitima"

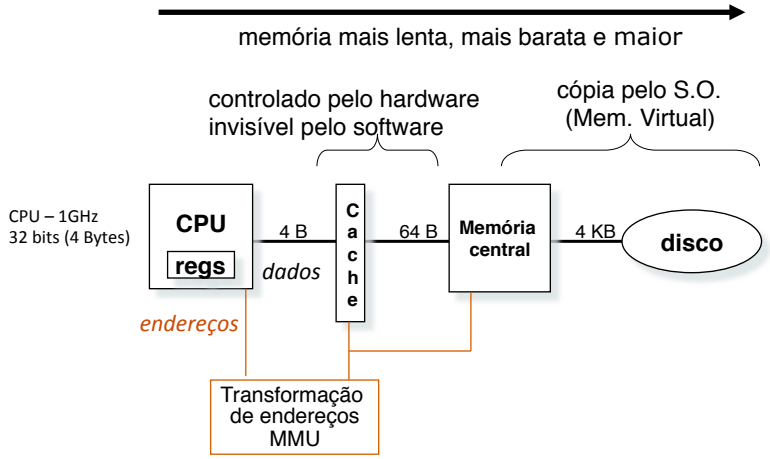
Transformação de endereços

V = {0, 1, . . . , N–1} espaço de endereçamento virtual
P = {0, 1, . . . , M–1} espaço de endereçamento físico

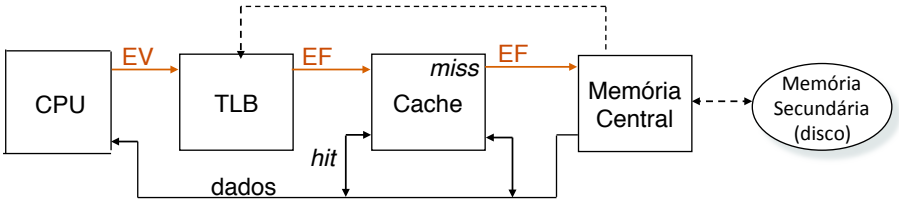
MAP(A) = A' se o dado no endereço A está presente no endereço físico A' (via TLB ou tab de páginas);
= *fault* se o endereço virtual A não está presente em memória



Exemplo de hierarquia de memórias

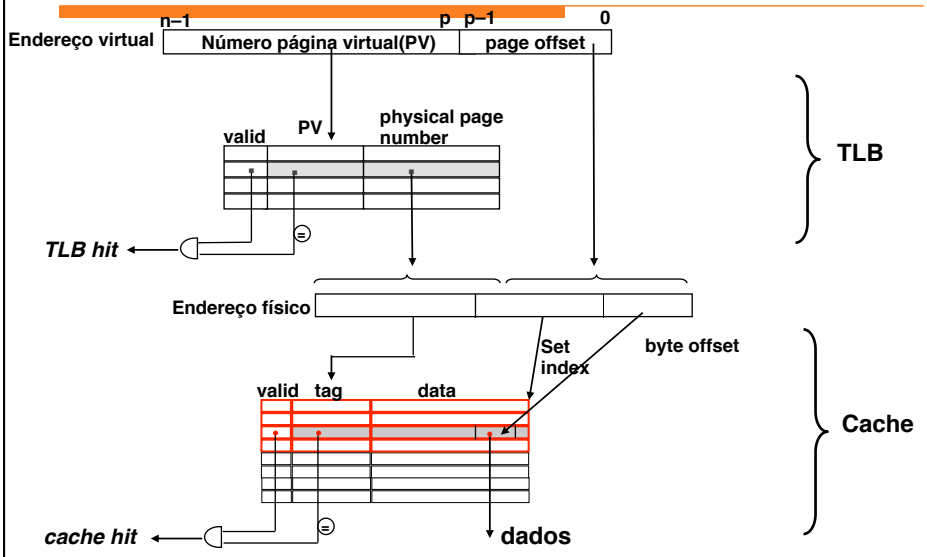


Integração da cache e da MV

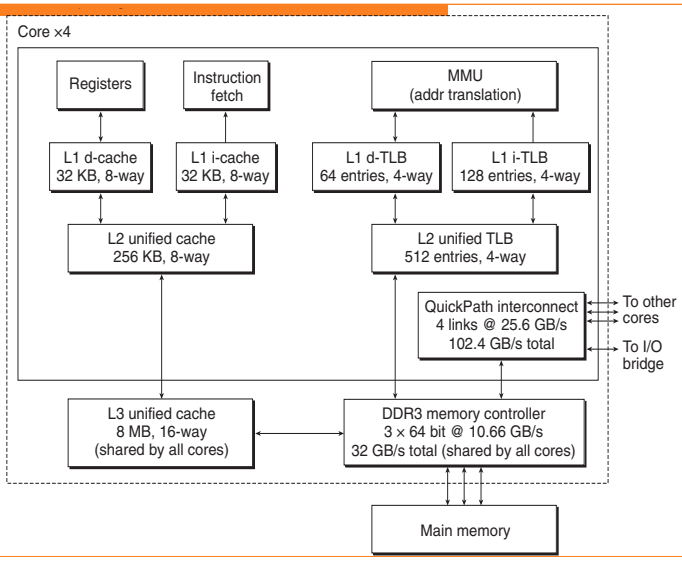


- A maior parte das caches são endereçadas pelo endereço físico
  - permite que diferentes processos tenham linhas na cache e que partilhem páginas
- Transformação de endereços antes do acesso à cache
  - Tal pode envolver acesso à RAM para consulta da tabela de páginas

Transformação de endereços (TLB e cache)

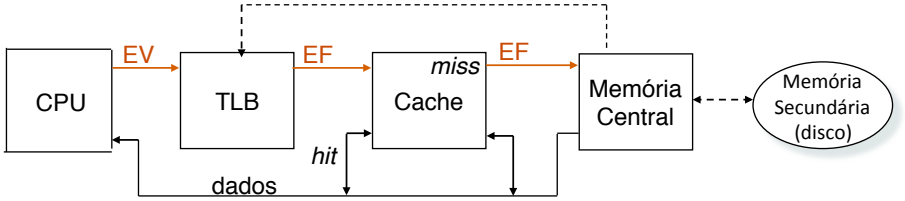


Exemplo Intel: caches e TLB (1 core)



Outro Exemplo

- Endereços: 32 bits
- Páginas: 4 KB
- Blocos: 32 Bytes
- L1: 64KB, 8 way set associative
- TLB: 64 entries, 4 way set associative



`mov Var, %eax`

- Resolvendo o endereço virtual:

`Var = 1314882 = 0x00141042 =`
`0000 0000 0001 0100 0001 0000 0100 0010`
`Páginas 4KB → num.pág=1314882/4K (4K=2^12)`
`20bits 12bits`
`00000000000101000001 000001000010`
***página 321** (0x141)*

- `TLB: 64/4 = 16 grupos (4 bits)`

`00000000000010100 0001` *`000001000010`*
`Tag=20 no grupo 1?`
`Senão vai à tab pág.`

Mov **Var, %eax**

- End virtual: Var = **1314882** = 0x00141042 = 00000000000101000001 000001000010
  página 321

- Assumindo: pág 321 → frame 96 (0x60)
- Então end real:
  00000000000001100000 000001000010
  = 0x00060042 = 393282

- Já pode aceder à memória (via cache)

End real: **mov (393282),%eax**

- Tratamento do endereço real:
  393282 **= 0x00060042 =**
  0000 0000 0000 0110 0000 0000 0100 0010
- dividindo de acordo com a cache:
  - 32B linha → 5bits
  - 64KB / (8*32B) = 256 grupos → 8bits
  - Tag (chave) → 32-13 = 19 bits

  00000000000000110000 00000010 00010

End real: `mov (393282),%eax`

- dividindo de acordo com a cache:
  - 32B linha → 5bits
  - 64KB / (8*32B) = 256 grupos → 8bits
  - Tag (chave) → 32-13 = 19 bits

`00000000000000110000  00000010  00010`

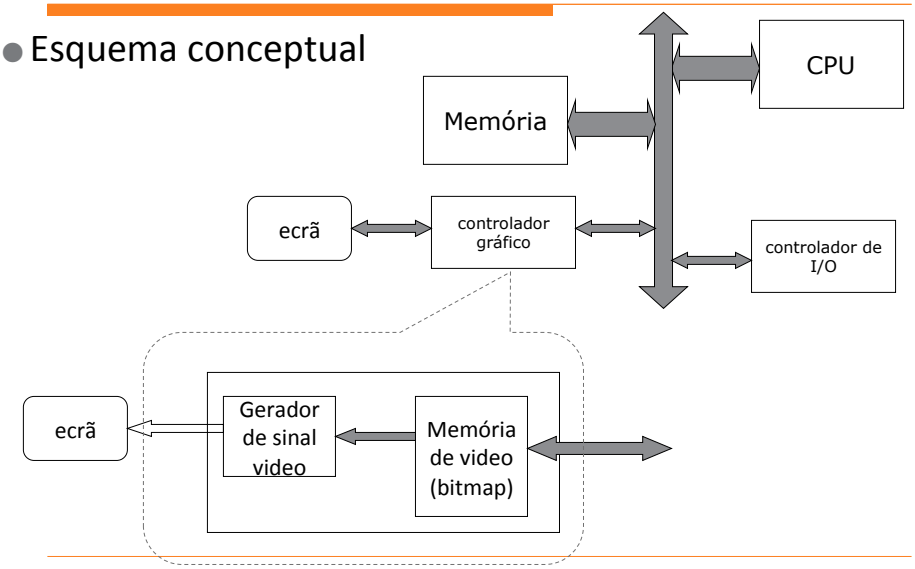
deve procurar no grupo 2
a tag = 48
se encontrar
lê a partir do byte 2
(4 bytes)

Graphical Workstation

- Antes: Gama de entre computadores pessoais (PC) e minicomputadores
- Um posto de trabalho (tipicamente um utilizador de cada vez), mas suportando múltiplos processos
  - Com capacidades gráficas, grande desempenho e capacidades de disco e memória
  - Normalmente com ligação a rede local para aceder a recursos remotos
- Agora: pode ser um PC

Arquitectura com consola gráfica

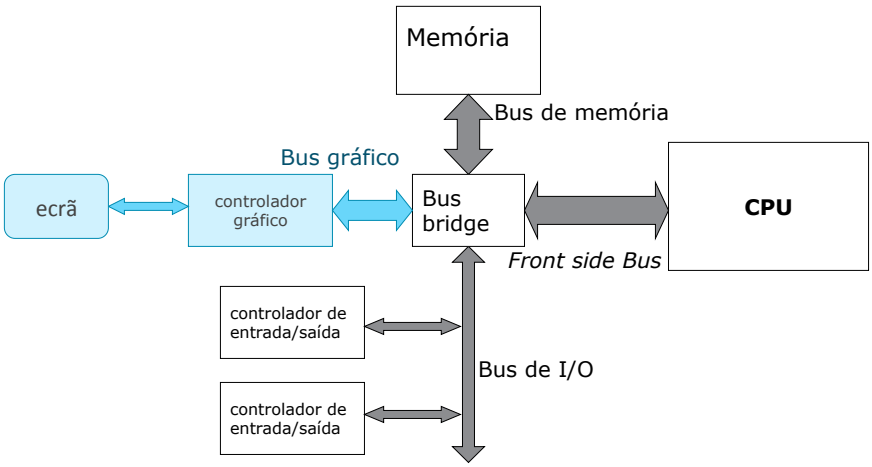
- Esquema conceptual



Transferências CPU-Mem. video

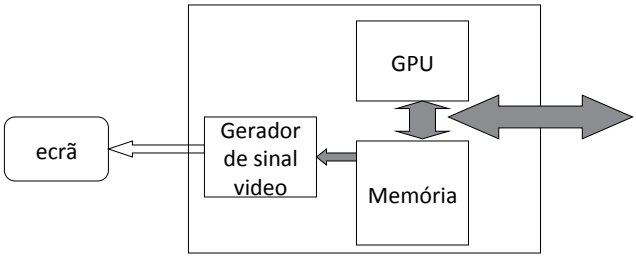
- Supondo uma resolução 1024x768, com 3 bytes de cor por pixel:
  - Memória vídeo = 3x1024x768 = 2,25 Mbytes
  - Para apresentar uma imagem completamente nova no ecrã é necessário transferir 2,25 MBytes
  - Se forem 20 imagens/s (20 frames/s) = 45 MB/s
- Grandes taxas de transferência, muita ocupação dos BUS e do CPU. Evolução:
  - O controlador gráfico é "promovido" na arquitetura
  - O controlador gráfico passa a incluir um coprocessador

Arquitectura com bus especial para gráficos



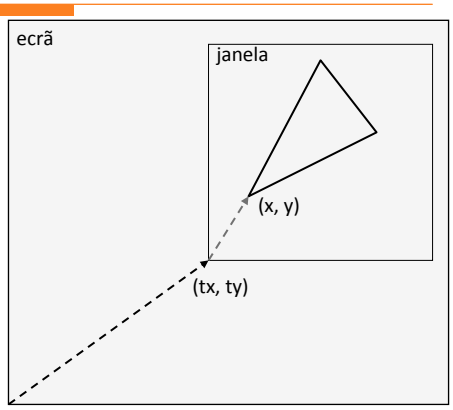
Coprocessador gráfico

- O controlador gráfico passa a ser "acelerado"
  - Inclui uma unidade coprocessadora que ajuda o CPU na manipulação de gráficos
  - GPU - **Graphics Processing Unit**



"Aceleração" 2D

- O GPU suporta comandos para desenho 2D
  - Em vez do CPU "pintar" os pixeis, manda a GPU fazer essas operações
    - Poupa o CPU e o BUS
  - Exemplos de comandos suportados: linhas, figuras geométricas (elipses, polígonos, etc), e também copia e move zonas da imagem…
  - Estas operações podem também incluir operações de translação, rotação e escala
    - Operações vectoriais



Posição do triângulo na janela: (x,y)
Posição da janela: (tx, ty)
Posição do trângulo no ecrã: (x+tx, y+ty)

Fases no desenho 3D

- Descrição da cena
  - os objectos 3D. Exemplo: por aproximação de triangulos 3D
  - Descrição das superfícies: Cor e textura
  - Descrição da cena: Posição dos objectos e da iluminação
- Projectar a cena 3D no plano do ecrã (2D)
  - Modelo da máquina fotográfica

  (resolver: posições de cada objecto e o aspecto das partes visíveis)

"Aceleração" 3D

- O GPU receba a descrição de toda a cena
- O GPU implementa um pipeline de operações:
  - Aplica todas as transformações nos objectos para os posicionar
    - Translações, rotações e escala
  - Calcula o "aspecto" de cada face, tendo em conta as características dos objectos e da luz que lhe incide
  - Efectua a transformação 3D para 2D, identificando as partes visíveis e calculando a cor de cada pixel no ecrã

Controladores gráficos – um segundo "computador"

- GPU quase/ou tão complexo como o CPU
  - Possui um *Instruction Set* bastante rico e eficiente (incluindo muitas inst vetoriais 3D)
  - Múltiplos processadores SIMD com centenas de unidades processadoras paralelas
  - GPU também chamado de GPGPU - *General Purpose Graphics Processing Unit*
- Têm bastante memória própria
- Exemplo: Nvidia GTX 1080
  - 2560 "cores" a 1,6 GHz
  - 8 GBytes de memória